
SONDERHEFT 94
Markt&Technik
öS 120,-/sfr 16,-/Lit 20000
hfl 21,-/dkr 75,-/fmk 72,-
DM 16,-
DISKETTE IM HEFT
64'er
Grafik
25 Programme auf Diskette
Char-Wandler
Supergrafik für fetzige Intros & Demos
Alan
Basic-Erweiterung für rasante Kurven, Linien und Kreise
Sprite Edit
32
Sprites für Action und Animation
Riesengrafik
Big Pic: 9 Scroll-Screens als Spielelandschaften
Großer Wettbewerb:
Wer programmiert den besten Trickfilm?

GEOS 2.5

Das neue ist das Beste, das es je gab.



Haben Sie sich schon immer gewünscht, Ihren C64 wie einen wirklich modernen Computer auszustatten? Mit ausgereifter Technologie und vielen nützlichen Utilities, mit denen die Arbeit leicht von der Hand geht? Dann können wir Ihnen das beste GEOS, das es je gab, anbieten: GEOS 2.5. Die brandheiße, neue Version bietet zahlreiche Erweiterungen und Überarbeitungen, die man einfach haben muß.

Neue Utilities:

- neuer Desktop mit Fenstertechnik und vielen neuen Funktionen
- Disk Turbo zur Verkürzung der Ladezeiten des Floppy-Laufwerks
- PrintDir druckt Ihre Directories
- neue NLQ-Druckertreiber
- 2 neue Maustreiber
- Silbentrenner nach Duden für GeoWrite-Texte

Neue Module:

- Foto- und Textmanager zum Einfügen von Text und Grafik in Ihre Dokumente
- GeoText, der schnelle Editor mit vielen GeoWrite-Funktionen
- ShowFont zeigt die Muster einer Schrift, bevor Sie sich entscheiden müssen

Aber das ist noch nicht alles, denn ein Wecker ist enthalten, ein Notizblock, ein Taschenrechner und und und ....
Mit dem besten GEOS geht jetzt eigentlich alles: Textverarbeitung, Bildverarbeitung, Desktop Publishing, Kalkulation, Präsentation, Telekommunikation, Dateiverwaltung ...
Machen Sie mit: GEOS 2.5 – das Beste, das es je gab. Jetzt im Fachhandel!

GEOS Version 2.5
Bestell-Nr. 57019, **DM 89,–***

Update auf Version 2.5
erhalten Sie direkt beim Markt&Technik Buch- und Software-Verlag GmbH & Co., Hans-Pinsel-Straße 9b, 85540 Haar bei München, gegen Einsendung der 1. Seite des Originalhandbuches und eines Verrechnungsschecks.

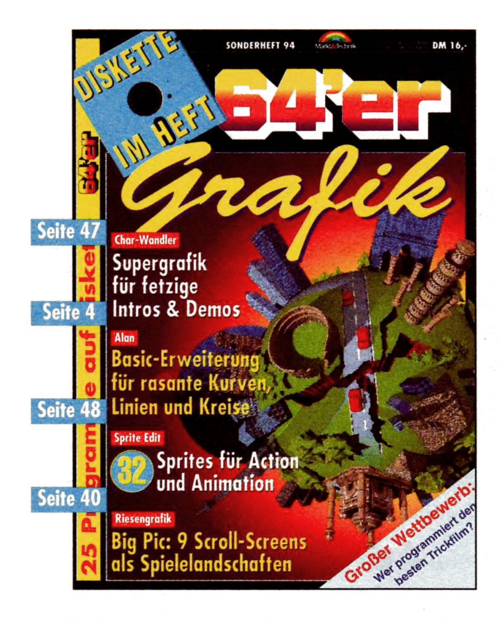
Update von 2.0 auf 2.5, Bestell-Nr. 57019A, DM 49,–*
Update von 1.X auf 2.5, Bestell-Nr. 57019D, DM 69,–*

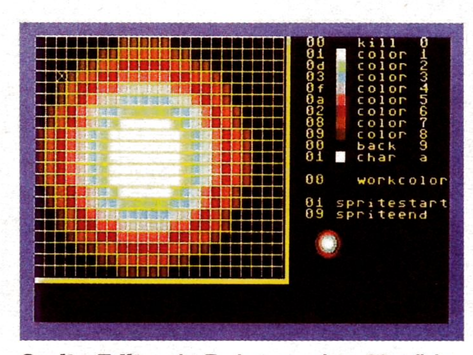
Das beste GEOS erhalten Sie bei Ihrem Buchhändler, in den Computerabteilungen der Warenhäuser und im guten Fachhandel.

* unverbindliche Preisempfehlung

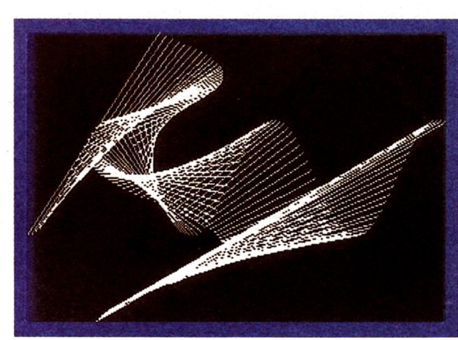








Sprite Edit: acht Farben machen 32 müde Sprites munter! **Seite 48**



Alan V7.2: eine der schnellsten Grafikerweiterungen für den C 64 **Seite 4**

Alle Programme aus Artikeln mit einem Disketten-Symbol finden Sie auf der beiliegenden Diskette (Seite 19)

Grafik im ICE-Tempo

Unsere raffinierte High-Speed-Grafik-Erweiterung macht's möglich: mächtige Befehle erzeugen die schönsten Grafiken, öffnen Bildschirmfenster oder aktivieren Screen-Scrolling.

Genau 43 neue Anweisungen: damit erweitert »Alan« den Basic-Interpreter des C 64, ohne das Basic-RAM allzu sehr einzuschränken – immerhin bleiben noch 34 KByte für Basic-Programme frei.

Neben den neuen Grafikanweisungen gibt's zusätzliche Editier-, Disketten-, Cursor-, Zeichensatz-, Scroll- und Window-Befehle – und alles in affenartiger Geschwindigkeit!

Laden Sie das Programm mit:

```
LOAD "LADER ALAN V7.2",8
```

und starten Sie mit RUN.

Die Hauptroutine »ALAN V7.2« und die übrigen Unterprogramme werden automatisch nachgeladen und aktiviert. Erscheinen Einschaltbild und Cursor, stehen die neuen Befehle zur Verfügung. Die Grafikerweiterung liegt resetfest im Speicher: weder <RUN STOP/RESTORE> noch der Reset-Knopf können sie aus dem Speicher verbannen.

Hier die Erläuterung der neuen Basic-Anweisungen (Kurzübersicht s. Kasten):

Diskettenbefehle

CAT

... zeigt das Directory der eingelegten Diskette im Laufwerk Nr. 8. Hält man <SPACE> gedrückt, stoppt der Listendurchlauf, bis man die Taste wieder losläßt. <RUN/STOP> bricht die Bildschirmausgabe ab. Das Basic-Programm im Speicher wird nicht gelöscht.

STATUS

... liest die Systemmeldung des Floppyfehlerkanals und gibt sie auf dem Bildschirm aus.

DISK "Befehl"

... schickt den gewünschten Disketten-Befehl ans Laufwerk. Man erspart sich die OPEN- und CLOSE-Anweisungen. Achtung: der Diskettenbefehl muß in Anführungszeichen stehen!

BLOAD "Filename"

... lädt ein Programm von Diskette, ohne irgendwelche Zeiger zu verändern (absolutes Laden). So lassen sich z.B. innerhalb eines Basic-Programms Assembler-Dateien laden, ohne das Basic-Programm erneut zu starten. Der neue Befehl ersetzt z.B. umständliche Programmiertricks wie:

```
10 IF A=2 THEN 100
20 IF A=0 THEN A=1: LOAD "Assembler-Pro-
gramm 1",8,1
30 IF A=1 THEN A=2: LOAD "Assembler-Pro-
gramm 2",8,1
100 REM weiter im Programm ...
```

Editierbefehle

HELP

... zeigt alle neuen Befehle als Liste auf dem Bildschirm.

OLD

Per »NEW« gelöschte Programme kann man unversehrt wieder zurückholen. Auch nach einem erweiterten Reset (z.B. bei modifiziertem Betriebssystem), nach dem sich Alan V7.2 mit SYS 49152 reinitialisieren läßt, kann man OLD einsetzen und ein eliminiertes Basic-Programm wieder aktivieren.

AUTO Anfangszeile, Schrittweite

... gibt die Zeilennummern (ab »Anfangszeile«) automatisch aus. Drückt man <RETURN> direkt hinter der Zeilennummer, wird der Modus aufgehoben.

REN Anfangszeile, Schrittweite

... vergibt neue Zeilennummern fürs gesamte Listing. Neunumerierung einzelner Bereiche ist leider nicht möglich. Die Aktion wird mit <RETURN> eingeleitet und läßt sich nicht mehr rückgängig machen. Schwachpunkt der Routine: Sprungbefehle (GOTO, GOSUB usw.) werden nicht angepaßt! Mit FIND (s. Beschreibung) kann man aber Basic-Zeilen mit solchen Sprunganweisungen schnell herausfiltern und von Hand begradigen.

DEL Anfangszeile - Endzeile

... löscht jede Basic-Zeile ab »Anfangs-« bis inkl. »Endzeile«. Zwischen beiden Parametern muß unbedingt der Bindestrich stehen. Die angegebenen Zeilennummern sollten tatsächlich existieren, sonst provoziert man die Fehlermeldung »Undef'd Statement Error«.

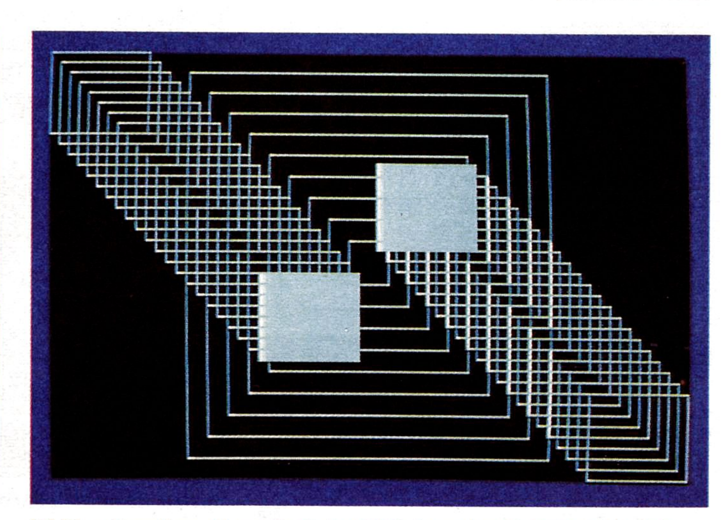
LLIST Anfangszeile - Endzeile

... schickt das Basic-Listing im Speicher zum Drucker mit der Geräteadresse 4. Verwendet man dieselben Parameter wie DEL, lassen sich Listing-Teile drucken. Ohne Angabe von Zeilennummern wird stets das gesamte Listing ausgegeben.

LSCROLL A

... scrollt Basic-Listings nach oben unten (sehr nützlich bei langen Quellcodes!). Per <F1> geht's aufwärts, <F3> scrollt nach unten. Hat die Variable »A« den Wert »1«, aktiviert man

[1] Farbige Balken-Grafiken per Block_Befehl



[2] Eine Kreation, die nur mit der REC-Anweisung erzeugt wurde

den Scroll-Modus; mit »0« stellt man ihn wieder ab. <F5> positioniert den Cursor bei aktivierter Anweisung in der linken unteren Ecke.

FIND Ausdruck

... durchforstet das Basic-Listing nach »Ausdruck« und bringt die Zeilen, in denen die Routine fündig wurde. Sie lassen sich unmittelbar danach bearbeiten. »Ausdruck« darf ein Basic-Befehl, ein Zeichen oder ein beliebiger String sein und muß direkt hinter dem Befehl stehen – also kein <SPACE> dazwischen! Anführungsstriche vor und hinter dem Suchbegriff sind unnötig.

Beispiel: »FINDSIN(X)« sucht überall nach dem Ausdruck »SIN(X)« im Basic-Listing. Die Ausgabe der gefundenen Zeilen kann man per <RUN/STOP>, nicht aber <RUN/STOP RESTORE>, unterbrechen.

CEN

... simuliert eine Centronics-Schnittstelle mit Parallelkabel am Userport

DMODE

... löscht nach <RETURN> die aktuelle Basic-Zeile, in der der Eingabe-Cursor steht.

DOFF

... macht DMODE wieder rückgängig: Ab sofort wird die Basic-Zeile nach <RETURN> wieder ins Programm integriert!

Cursor-Positionierungsbefehle

HOME A

Der Parameter »A« kann »0« oder »1« sein:

– **HOME 0:** löscht den Screen und plaziert den Cursor an den Koordinaten 0/0 (linke obere Ecke).

– **HOME 1:** löscht den Bildschirm ebenfalls, bringt aber den Cursor auf Position 0/22. Hier beginnt beim Split-Screen (s. Grafikmodus 1) das Textfenster; mit »PRINT« kann man direkt ins Window schreiben.

VTAB A

Damit schickt man den Cursor zur Zeile »A« (erlaubte Werte: zwischen 0 und 24).

Beispiel: VTAB 5 setzt den Cursor an den Anfang der fünften Bildschirmzeile.

Zeichensatz-Manipulationen

CCOPY

... kopiert den C-64-Original-Zeichensatz ab Adresse $D000 (53248) im ROM in den RAM-Bereich ab $3000 (12288), wo er sich nach Herzenslust ändern läßt.

CNEU

... aktiviert die Zeichensatz-Muster ab $3000.

CALT

Der C 64 holt den Zeichensatz wieder aus dem Originalbereich im ROM ab $D000.

Scroll-, Window-, Farbbefehle

WGET D$

... erlaubt die Eingabe der Zeichenkette A$. Das macht auch der INPUT-Befehl, aber: jetzt sind auch bislang »verbotene« Zeichen wie Semikolon, Komma und Anführungszeichen gestattet. Die Eingabe wird im Kassettenpuffer ab Adresse $033C (828) zwischengespeichert. Dieser String läßt sich dann vom SCROLL-Befehl weiterverwenden, muß jedoch stets neu geschrieben werden, da die GET-Routine des Betriebssystems aktiv ist.

TRANSFER A$

... speichert die Zeichenkette A$ direkt im Kassettenpuffer. Nachfolgende SCROLL-Anweisungen können diesen String weiterverwenden (wenn man z.B. eine Laufschrift erzeugen will, ohne den Text über die Tastatur eingeben zu müssen!).

SCROLL A

»A« enthält die Anzahl der Zeichen (von 1 bis 255), die in den beiden untersten Bildschirmzeilen von rechts nach links scrollen. Die Routine holt sich den Text aus dem Kassettenpuffer, wo er mit TRANSFER oder WGET abgelegt wurde. Es empfiehlt sich, am Anfang der Zeichenkette zwei Leerzeichen freizulassen. Die Anweisung TEXT beendet den Scroll-Modus.

WINON X, Y, Länge, Breite, Nummer

... definiert ein Bildschirmfenster. X und Y sind die Anfangskoordinaten (linke obere Ecke); »Länge« und »Breite« bestimmen die Window-Größe (Koordinatenwerte des Textbildschirms!). »Nummer« akzeptiert nur die Zahlen »1« oder »2« und gibt an, welches Window eingeschaltet wurde. Achtung: Aus Speicherplatzgründen lassen sich nicht mehr als zwei Windows definieren!

Wenn das Fenster auf dem Bildschirm erscheint, füllt sich der definierte Bereich zunächst mit inversen Leerzeichen. Die Farbe bestimmt man mit »POKE 39169, Farbwert«. Der Cursor setzt sich im Window fest und wird durch eine programminterne Abfrage daran gehindert, es wieder zu verlassen.

WINOFF

... schaltet das Fenster wieder aus: die alten Werte werden zurückgeholt und zusätzlich der Interrupt wieder auf normale Konfiguration eingestellt. Jetzt kann man den Cursor wieder frei über den gesamten Screen bewegen.

COL Rahmen-, Hintergrund-, Zeichenfarbe

... setzt die gewünschten Bildschirmfarben.

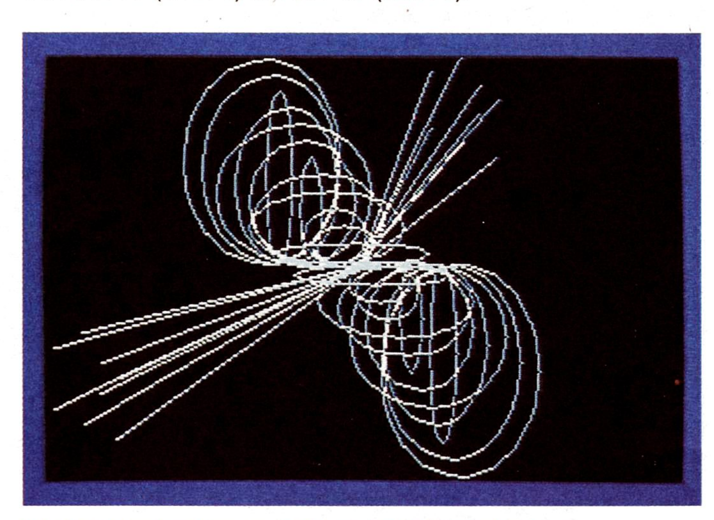
Grafikanweisungen

HGR A

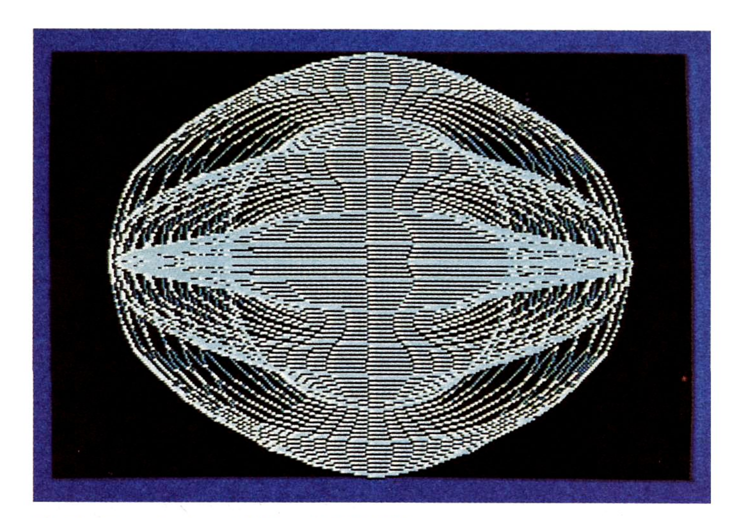
»A« kann den Wert 0 oder 1 haben:

– HGR 1: schaltet Hires-Grafik für den gesamten Bildschirm ein (Auflösung: 320 x 200 Pixel). Die Bitmap liegt im Bereich von $6000 (24576) bis $7F40 (32576).

[3] Grafikbeispiele mit REC- und CIRCLE



[4] Eine Kombination der REC- und BLOCK-Anbindung



[5] Grafikvarianten: LINE und CIRCLE

– HGR 0: verwendet den oberen Bereich des Bildschirms als Grafikfenster; die unteren fünf Zeilen sind als Textbereich vorgesehen. Mit »HOME 1« löscht man den Textbildschirm und positioniert den Cursor bei Spalte 0, Zeile 20. Diese Betriebsart eignet sich ideal für Adventures (oben Grafik, unten Text). Die Grafikbitmap beginnt ebenfalls bei $6000; besitzt aber nur noch eine 320 x 168-Punkte-Auflösung.
Achtung: Im Modus »HGR 0« kann man weder Windows definieren noch die Laufschrift (SCROLL) einschalten, da alle drei Routinen den Interrupt-Zeiger verbiegen.

TEXT

... stellt die Grafik ab, aktiviert wieder den Textbildschirm und stoppt eventuell eingeschaltete Laufschriften.

GCLR

... löscht den Hires-Screen.

SET Zeichen-, Hintergrundfarbe

... legt die Farben des Hires-Grafikbildschirms fest.

MODE A

... stellt den Zeichenmodus ein (»A« ist entweder 0 oder 1):
– MODE 0: teilt der Routine mit, alle Grafikpunkte zu setzen.
– MODE 1: löscht alle Grafikpunkte.

INV

... invertiert den Grafikbildschirm (EOR-Verknüpfung): vorher gesetzte Punkte werden gelöscht, inaktive dagegen eingeschaltet.

PLOT X, Y

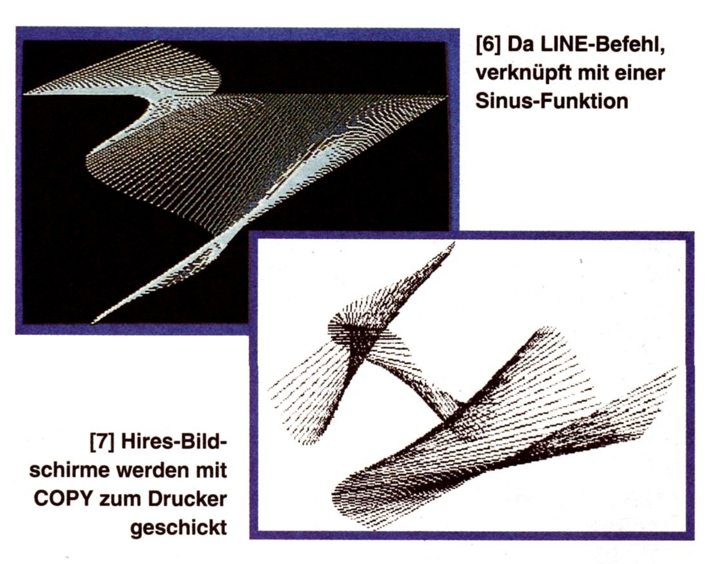
... schaltet Bildpunkte an der x- und y-Position des Grafikbildschirms ein. Die x-Koordinate darf den Wert 319, die Y-Koordinate »199« nicht überschreiten, sonst erscheint eine Fehlermeldung.

Beispielprogramm für eine Sinuskurve:

```
10 HGR 0        : REM Grafik ein (gesplittet)
20 GCLR                    :REM Grafik löschen
30 SET 3,0                   :REM Farbe setzen
40 HOME 1          :REM Textbildschirm löschen
50 PRINT "Sinuskurve"
60 FOR X = 0 TO 319
70 PLOT X,SIN(X/5)*20+100
80 NEXT
90 GET A$ : IF A$ = " " THEN 90
100 HOME 0              :REM Bildschirm löschen
110 TEXT :REM Auf Textbildschirm umschalten
120 END
```

LINE X1,Y1,X2,Y2

... zieht eine Linie im Grafikbildschirm. Die Koordinaten X1 und Y1 bezeichnen den Anfangspunkt, X2 und Y2 den Endpunkt. Wie beim PLOT-Befehl gelten auch hier die Einschränkungen der Koordinatenwerte (x = maximal 319, y = maximal 199).



[6] Da LINE-Befehl, verknüpft mit einer Sinus-Funktion

[7] Hires-Bildschirme werden mit COPY zum Drucker geschickt

HLINE X, Y, Länge

... gehört zu den stärksten Befehlen von Alan V7.2: horizontale Linien entstehen in Sekundenbruchteilen. Bedenkt man, daß die meisten Linien gerade verlaufen, kann man sich eine Eigenart der Alan-Bitmap zunutze machen: ein gesetztes Byte entspricht einer horizontalen Linie von acht Punkten. Jetzt berechnet man den Anfangspunkt und teilt die Linienlänge durch »8«. Anschließend addiert man »8« zur Anfangsadresse und setzt dieses Byte. Dieser Vorgang läßt sich so oft wiederholen, bis die Anzahl der Schleifendurchläufe exakt »INT(Länge/8)« beträgt. Die restlichen Pixel (die vorderen und hinteren acht) kann man mit der normalen LINE-Routine setzen. Dieser Algorithmus hat den immensen Vorteil, daß nicht jeder einzelne Punkt berechnet und gesetzt werden muß (ist bei den meisten Grafikerweiterungen der Fall!).
Die Parameter x und y definieren den Anfangspunkt; »Länge« ist die Ausdehnung der Linien in Bildpunkten. Man sollte aber auf jeden Fall darauf achten, daß »x + Länge« den Wert 319 niemals übersteigt!

VLINE X, Y, Länge

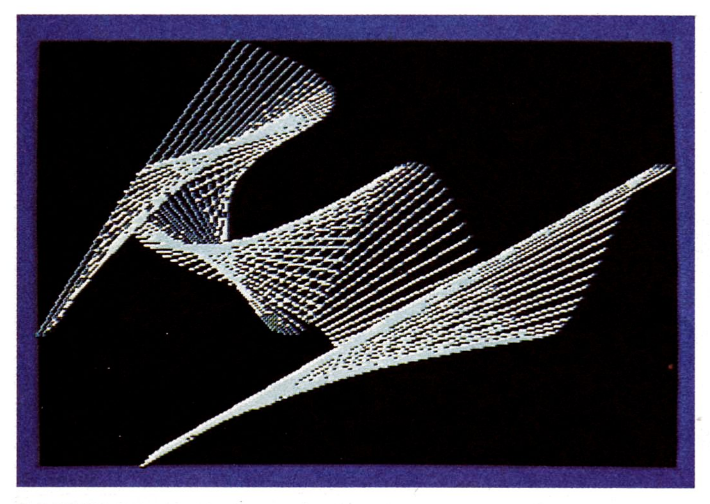
... entspricht HLINE, zieht aber vertikale Linien. Die faszinierende Geschwindigkeit ist geblieben. Auch hier rekrutiert

Druckertips

serieller Anschluß: Die Druckerroutinen von Alan V7.2 arbeiten mit Epson-kompatiblen Druckern, die Sekundäradresse »1« benutzen. Sollte Ihr Gerät oder das serielle Hardware-Interface einen anderen Wert brauchen, müssen Sie die entsprechenden Speicherstellen nach dem Laden von Alan V7.2 ändern:
– Textdruck (LLIST): POKE 53140, Sekundäradresse
– Grafikdruck (COPY): POKE 37220, Sekundäradresse
Parallelkabel am Userport: Mit CEN aktiviert man die eingebaute Centronics-Schnittstelle, die ein Parallelkabel am Userport braucht. Hier sind die Sekundäradressen ohne Belang. Eventuell müssen Sie am Drucker den entsprechenden DIP-Schalter für Zeilenvorschub (LF) auf ON stellen!

Alan V7.2 (Befehlskurzübersicht)

CAT	BLOAD	CCOPY	CNEU
CALT	HELP	HGR	INV
GCLR	TEXT	STATUS	DISK
TRANSFER	WGET	COL	VTAB
SET	LLIST	FIND	OLD
GSAVE	CHANGE	DEL	DMODE
DOFF	SCROLL	CEN	AUTO
INK	WINON	WINOFF	PLOT
HLINE	VLINE	BLOCK	REC
CIRCLE	HOME	LSCROLL	REN
MODE	LINE	COPY	



[8] CIRCLE-Befehl in Sinus-Kombination

sich die Startposition aus X und Y, in Bytes umgerechnet und der Punkt gesetzt. Addiert man »1«, erhält man den nächsten zu setzenden Punkt. Ist ein 8-Byte-Block zu Ende, zählt man »312« dazu, um zum nächsten zu kommen. Auch hier gilt: »y + Länge« darf »199« nicht übersteigen!

BLOCK X, Y, Länge, Breite

... zeichnet ein ausgefülltes Rechteck (Abb. 1). X und Y definieren die linke obere Ecke; »Länge« und »Breite« den Umfang des Blocks (Offset = relativ zur Position von x und y). Da diese Anweisung ebenfalls HLINE-Routine benutzt, sollten Sie die Parameter sorgfältig wählen! Wir kennen keinen schnelleren BLOCK-Befehl für den C 64: er plaziert ca. 128 000 Pixel pro Sekunde; 64000 Punkte (also ein Hires-Screen) ist in 27/60 Sekunden gefüllt. Probieren Sie's aus:

```
10 HGR 1: GCLR
20 BLOCK 0,0,319,199
```

REC X, Y, Länge, Breite

... ist mit BLOCK gleichzusetzen (Parameter haben die gleiche Bedeutung); allerdings zeichnet der Befehl nur die Verbindungslinien des Rechtecks (Abb. 2).

CIRCLE X, Y, X-Ausdehnung, Y-Ausdehnung

... bringt beliebige Ellipsen auf den Hires-Screen. X und Y sind die Mittelpunkt-Koordinaten; »X-« und »Y-Ausdehnung« die Radien in der x- und y-Achse. Sind die Werte gleich, entsteht ein Kreis, sonst Ellipsen. Bedenken Sie bei der Parameterwahl: Der Kreisradius wirkt im 360-Grad-Winkel. Der 180-Grad-Radius errechnet sich also aus »y + y-Ausdehnung« und darf den Maximalwert von »199« nie überschreiten; der Radius von 0 Grad dagegen wird aus »y - y-Ausdehnung« berechnet und darf nicht kleiner als »0« sein (sonst bricht die Routine ab). Experimentieren Sie mit diesem Beispiel:

```
10 HGR 1: GCLR
20 CIRCLE 160,100,99,99
1000 POKE 198,0: WAIT 198,1
1010 TEXT
```

INK neue Farbe, alte Farbe

... malt den ansonsten einfarbigen Grafikbildschirm bunt. Die Auslösung von 320 x 200 Punkten bleibt dennoch erhalten – also nicht zu verwechseln mit dem Multicolormodus. Es wird in jedem 8-Byte-Block geprüft, ob ein Pixel gesetzt ist. Falls nicht, ändert das Programm den entsprechenden Wert im Video-RAM. Ist der Punkt gesetzt, bleibt der alte Farbwert bestehen. Das verhindert aber nicht, daß die übrigen Punkte des 8-Pixel-Blocks in der per COL-Anweisung definierten Hintergrundfarbe erscheinen. Grund: der C 64 besitzt zu wenig Speicherplatz, um für jedes Hires-Pixel die eigene Farbinformation zu speichern – das geht nur als Byte (acht Bildpunkte).

COPY

... schickt den Inhalt des Grafikbildschirms ($6000 bis $7E40) als Hardcopy zum Drucker. Für die Besitzer eines Userport-Centronic-Parallelkabels ist ein Treiber integriert, den man mit »CEN« aktiviert. Wenn Sie den COPY-Befehl verwenden, muß der Grafikbildschirm ausgeschaltet sein (TEXT), sonst kommt's zu Fehlfunktionen.

CHANGE A

... tauscht Grafikspeicher untereinander aus:

– A = 0: wechselt den Grafikbereich von $6000 bis $7E40 mit dem von $A000 bis $BE40.

– A = 1: OR-Verknüpfung der Bitmap ab $6000 mit der ab $A000. Das neue Bild wird wieder im Speicher ab $6000 abgelegt.

GSAVE "Filename",8

... sichert den Bereich von $5C00 bis $8000 mit beliebigem Dateinamen auf Disk. Es werden also das aktuelle Video-RAM ($5C00 bis $5FFF) und zugleich die Bitmap ($6000 bis $7FFF) gespeichert – damit die Farben beim späteren Laden wieder stimmen! Nachteil: Die Grafik liegt nicht im Standardbereich von $2000 bis $3E40 (Hi-Eddi-Standard) und muß erst dorthin übertragen werden, wenn man sie mit Zeichenprogrammen nachbearbeiten will:

```
FOR I=0 TO 7999: POKE 8192+I,
PEEK(24576+I): NEXT
```

Soweit die neuen Befehle, die jedem Basic-Programmierer wertvolle Dienst leisten, der sich mit der zeitkritischen hochauflösenden Grafik auseinandersetzt.

Demoprogramme auf Diskette

Um Ihnen vor Augen zu führen, was die neuen Anweisungen können, finden Sie fünf kurze Demoprogramme auf der Diskette zu diesem Sonderheft. Sie lassen sich wie jedes Basic-Programm laden, aber nur im vorher aktivierten Modus der Basic-Erweiterung Alan V7.2 starten:

– **DEMO1:** ... unterstreicht den Speed, mit dem der REC-Befehl arbeitet, auf einem geteilten Bildschirm (Split-Screen). Die Eingabe von TEXT im unteren Fenster stellt den Normalzustand wieder her.

– **DEMO2:** ... ist eine gelungene Mixtur von REC-, LINE- und PLOT-Anweisungen,

– **DEMO3:** ... zeigt, wie schnell unterschiedlich gefärbte Rechtecke (quasi als Balkengrafik) auf dem Screen erscheinen. Das untere Textfeld eignet sich zur Beschriftung der Grafik.

– **DEMO4:** ... ist eine Sammlung von Grafikbeispielen und erzeugt die Abbildungen 3 bis 8 in faszinierender Geschwindigkeit.

– **DEMO5:** ... ist eine Demonstration der Floppybefehle von Alan V7.2 und weiteren Grafikbeispielen. Raffiniert: der Einsatz des WINON-Befehls zur Ausgabe von Benutzerhinweisen!

Alan V7.2 eignet sich aufgrund der blitzschnellen Ausführung der Grafikbefehle und seines Split-Screen-Modus ideal zum Programmieren raffinierter Grafik-Adventures oder für Chart-Programme, die trockene Zahlen in Balken, Torten oder Kurvengrafik verwandeln. Beachten Sie unsere Druckertips im Textkasten. (bl)

Kurzinfo: Alan V7.2

Programmart: Grafikerweiterung
Laden: LOAD "LADER ALAN V7.2",8
Starten: nach dem Laden RUN eingeben
Besonderheiten: sehr schnelle Routinen zum Erzeugen von Linien, Rechtecken und gefüllten Flächen
Benötigte Blocks: 31
Programmautoren: Andreas Pustelny/Alexander Gaß


Wow! Soviel
Top Themen
64'er
Sonderhefte
Sonderhefte zum Nachbestellen
Jetzt zum Preis von DM 16.- je Heft,
128er Sonderhefte für DM 24,-.
Ordnung im Archiv mit der praktischen Sammelbox für nur DM 14,-.
Einsteiger
SH 50: Starthilfe
Alles für den leichten Einstieg/ Heiße Rythmen mit dem C64/ Fantastisches Malprogramm
SH 62: Erste Schritte
Exbasic Level II: Die Super-Basic-Erweiterung RAM-Exos: Floppy-Speeder der Extraklasse
SH 74: Einsteiger
Basic 3.5.: Basic-Erweiterung mit Hires-Grafik-Befehlen / FOBS V1.0: Floppy Operationen per Benutzeroberfläche
Anwendungen
SH 68:
Faszination Sternenhimmel. Der Blick in den Kosmos/ Kreuzworträtsel zum Selbermachen
SH 78:
Grafic-Calc:Grafische Auswertung von Jahresbilanzen/MAS V1.0: Übersichtliche Schulnotenverwaltung
SH 81:
Paint Mania: Zeichenprogramm der Superlative/ Maestro: Eigenen Sound auf Knopfdruck komponieren/ Disk Tools V6.5: Enttarnt jedes Byte auf Diskette
SH 86:
Database 2.0: Universelle Datenbank mit starken Rechenfunktionen/ Stamp Collection Kit: Archiv für Ihre Briefmarken/ außerdem: Autokauf, Haushaltskasse & Girokonto
GEOS
SH 48: GEOS-Erweiterungen
Geotext - neuer, schneller Texteditor für Geowrite/Workshop zu Geopublish
SH 59: GEOS
GeoBasic: Großer Programmierkurs mit vielen Tips & Tricks
SH 80: GEOS
Lottoblock: Statistische Gewinnzahlauswertung mit Tippvorschlägen/ Finanzen: Welche Geldanlage ist die Beste ?
Tips,Tricks & Tools
SH 65:
Streifzug durch die Zeropage/ Drucker-Basic: 58 neue Befehle zur Printer-Steuerung/ Multicolorgrafiken konvertieren/ über 60 heiße Tips&Tricks
SH 77:
Amica-Konvert: 6 Malprogramme tauschen Grafik aus/ Disc-Basic: Floppybefehle kurz und prägnant prüfen
Hardware
SH 67:
Wetterstation.Temperatur, Luftfeuchtigkeit und -druckmessen/ DCF Funkuhr und Echtzeituhr/ Daten konvertieren: vom C64 zum Amiga, Atari ST und PC
SH 83: Floppy
4 Kopiertools für komplette Backups und Einzel-Files/ Spur 36 bis 40: 95 Blocks zusätzlich auf Diskette viele Tips&Tricks
SH 84: Hardware
C64 kompakt: Computer und Floppy in einem Gehäuse (Umbauanleitung)/ Midi-Interface: C64 mit Keyboard/ Tiny-EPROMer: EPROM-Brenner im Selbstbau unter 30 DM
Diskette im Heft
Hardware
Floppy-Speeder für 10 Mark
8-Kanal-Schaltinterface
Sieben Sinne für den C 64
Der Edel-C 64 mit Floppy
10 Bauprojekte mit Programmen
Spiele
11 Superspiele für Drahtseilnerven
Daten-Agent 00X: Jagd im All
Zoff im Geisterschloß
Charakter-Editoren: Spielelösungen, Pokes und Cheats
Alle Programme auf Diskette
Alle Hefte mit randvoller Programm-Diskette
Top Spiele mit Diskette
Action
Jump'n Run
Logik

Know how! plus Diskette !

Programmier-Sprachen

SH 71: Assembler
Kursus/ Komplettpaket/ Befehlsposter/ Tips & Tricks/ Leserfragen

Grafik

SH 55:
Amica Paint: Fantastisches Malprogramm für Hobby-Grafiker, mit allen Up-Dates

SH 75:
Interlace 64: 136 Farben und 640x200-Pixelgrafik und 80-Zeichen-Bildschirm/ 3D-Animation mit Hi-Eddi

Drucker

SH 72:
Publish 64: DTP-Einstieg/ Topprint: Druckt Briefköpfe, Schilder und Grußkarten/ Test: Drucker unter 1.000 DM

C 128

SH 58:
Übersichtliche Buchhaltung zuhause/Professionelle Diagramme

SH 64: Anwendungen:
Amerikanisches Journal/ Grundlagen: CP/ M, das dritte Betriebs-system/ VDC-Grafik: Vorhang auf für hohe Auflösung

SH 76:
Music Master: Professionelle Datenbank zur LP- und CD-Sammlung/ Prüfungsaufgaben: Idealer Helfer vor jedem Examen .

SH 82:
Mini-Micro: Kopiert 1571-Disketten zur 1581/ Pro-Book 128: Komfortable Datenbank für den Bücherwurm

Spiele

SH 2: Top Spiele
10 Game-Trainer und 2 Longplays/ Katakis-System: Spiele programmieren wie die Profis/ Tolle Tips für höhere Level und Unsterblichkeit

SH 3: Top Spiele
Action Jump'n Run Logik/ Tips, Tricks & Tools

SH 54:
15 tolle Spiele auf Diskette/ Der Sieger unseres Programmierwettbewerbs: Crillion II/ Ein Cracker packt aus: Das ewige Leben bei kommerziellen Spielen

SH 60: Adventures
8 spannende Abenteuerspiele/ 2 Komplettlösungen und viele Tips, Adventures selber programmieren

SH 61:
20 heiße Super-Games für Joystick-Akrobaten/ Cheat-Modi und Trainer POKEs zu über 20 Profi-Spielen/ Krieg der Kerne: Grundlagen Spieleprogrammierung

SH 66:
15 Top-Spiele mit Action und Strategie/ Mondlandung: Verblüffend echte Simulation und Super-Grafik/ High-Score-Knacker: Tips & Tricks zu Action-Games

SH 73:
Action bis Adventure: 10 Spiele zum Kampf gegen Fabelwesen/ Preview/ Tips & Tricks/ Kurse/ Game Basic/ Mission II/ W.P.Tennis II/ Omnibus GmbH/ Mic's Push'em

SH 79:
25 starke Mega-Games/ Exis: Gefährlicher Satelliten-schleppdienst/ Tips & Spielelösungen

SH 85:
11 Super-Games für stählerne Nerven/ Datenagent 00X: Noch 12 Stunden bis zum Weltuntergang/ Kick'n kill: Irrwitzigees Jump-and-Run-Spiel für Joystick-Akrobaten

Nur noch hier erhältlich!

Jetzt sofort Bestellen - per Post oder FAX !

Ordnung im eigenen Archiv für DM 14.-

Hier war leider jemand schneller, doch null Problem: Einfach schreiben und bestellen bei 64er -Magazin Leserservice, 74170 Neckarsulm, Telefon 07132/969-185, FAX: 07132/969-190

BESTELLCOUPON

Ich bestelle folgende 64er Sonderhefte:

	SH-Nr.	SH-Nr.	SH-Nr.	
____Sonderhefte mit Diskette je 16,- DM				DM
____Sonderhefte"128er" je 24.-DM				DM

Ich bestelle______Sammelbox(en) ____DM

zum Preis von je DM 14,- **Gesamtbetrag** ____DM

Ich bezahle den Gesamtbetrag zzg. Versandkosten nach Erhalt der Rechnung.

Vorname,Name

Straße, Hausnummer

PLZ, Wohnort

Datum / Unterschrift

Schicken Sie bitte den ausgefüllten Bestellcoupon an:
64er-Magazin Leserservice, 74170 Neckarsulm, Telefon: 07132/969-185
oder bequem per Telefax: 07132/969-190

Tolp 64 – Grafikwunder

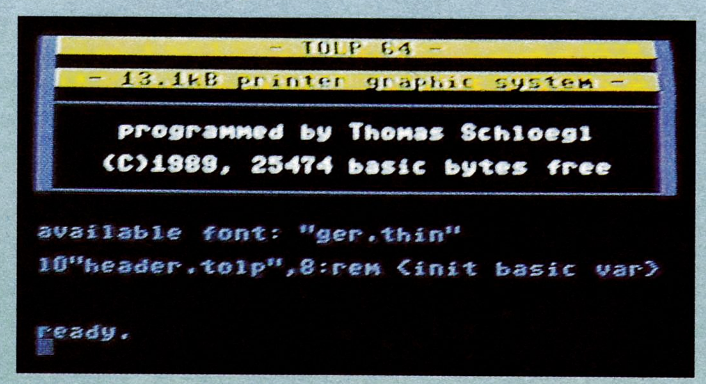
Drucker als Grafikbildschirm

Faszinierende Bilder, Schriften und Vektorgrafik: das gab's in dieser Ballung noch nie für den C 64 mit Drucker. Unsere Basic-Erweiterung »Tolp 64« aber macht's möglich.

Hochauflösende Grafik läßt sich direkt zu Epson-kompatiblen Druckern schicken, die seriell mit dem C 64 verbunden sind – ohne Umweg über den Bildschirm.
Laden Sie das Programm mit:

```
LOAD "TOLP 64",8
```

Gestartet wird mit RUN.
Der Computer holt nun den Default-Zeichensatz (»ger.thin«) und die Initialisierungs-Routine (»header.tolp«) in den Speicher. Wenn sich der Startbildschirm mit READY wieder meldet (Abb. 1), stehen alle neuen Grafikdruck-Befehle zur Verfügung.

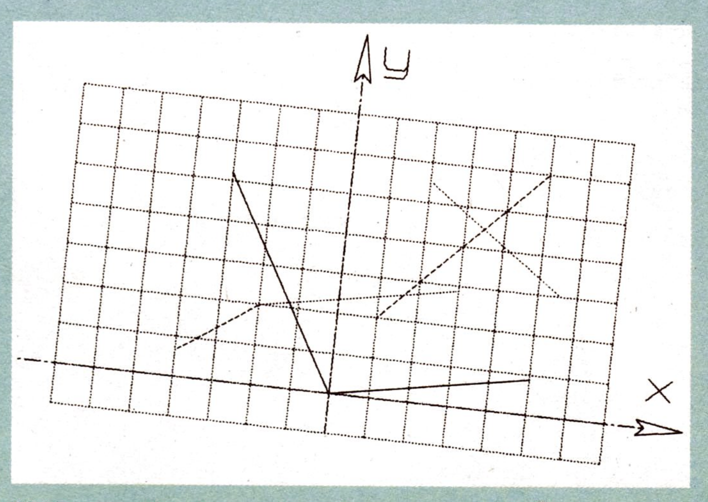


[1] Das Hauptmenü von Tolp 64: komplexe Befehlserweiterung für anspruchsvolle Grafiken

Linien, die man zeichnen will, definiert man zunächst per PLOT-Anweisung und speichert sie im »Main-Array« (Hauptfeld). Die Linien lassen sich per EXEC oder automatisch (bevor das Main-Array überläuft) ausdrucken. Das ist das Kernstück dieser Befehlserweiterung. Der Grafikdruck funktioniert nur bei Epson-kompatiblen Druckern mit der Geräteadresse 4 (s. Kasten »Druckerhinweise«).

Kapazität des Hauptfeldes

Das Main-Array kann maximal 255 Linien zwischenspeichern. Spätestens dann geht's los mit dem Ausdruck. Selbstverständlich lassen sich auch Grafiken ausgeben, die sich aus weniger Linien zusammensetzen (»single graphic«).
Zusätzlich kann man elegant auch komplizierte Grafiken zeichnen. Verwenden Sie dazu einfach ein Blatt Traktorpapier (DINA4) und kleben es an Anfang- und Endkante so zusammen, daß eine Endlosrolle entsteht (natürlich bereits bei eingelegtem Papier und montierter Traktorführung). Damit simuliert man quasi einen plotterähnlichen Betrieb (»plotter graphic«). Achtung: das Klebeband sollte äußerst sorgfältig angebracht werden. Vor allem darf kein Raum zwischen den beiden Papierenden bleiben – sonst wird die Traktorwalze verschmutzt und stoppt die Papierrolle beim Transport.
Probleme gab's bei unserem Test nur, wenn der Rand des Klebebandes mit der Zeit (nach ca. 30 Seitenvorschüben) begann, sich zu lösen. Daher sollte man bei Grafiken sicherheitshalber den Seitenanfang markieren und das Programm unterbrechen, wenn das Blatt beginnt, sich zu verschieben. Bei aktivierter Interrupt-Routine (nach <RUN/STOP RESTORE> unbedingt den Befehl INIT eingeben!) kann man das Programm durch gleichzeitigen Dauerdruck auf die CTRL- und CBM-Tasten stoppen, den Druckkopf auf die Markierung setzen und diese Position wieder als Seitenanfang definieren (notfalls: Drucker aus- und einzuschalten). Die zu druckenden Daten werden in den Bereichen $A000 bis $BFFF und $E000 bis $FFFF (RAM unterm ROM) aufbereitet. Je nach Druckdichte kann das ziemlich lange dauern (s. BOUND-Befehl): bedenken Sie die Datenmenge die für 320 Pixelspalten und 200 Rasterzeilen zu berechnen sind!



[2] Vektorgrafiken in x- und y-Richtung: eine leichte Übung

Unverzichtbar: Arrays

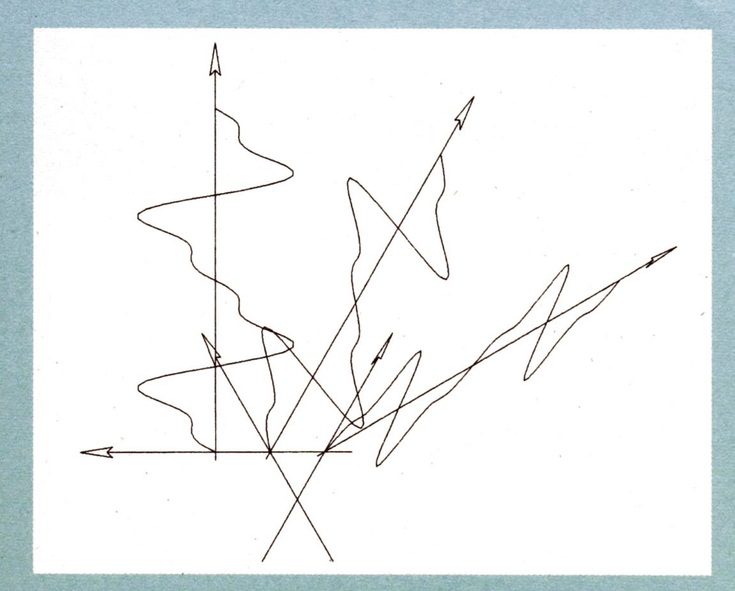
Erforderliche Arrays (Variablenfelder) dimensioniert man mit den DEFINE-Anweisungen (s. DFMAIN und DFSUB). Intern wandelt sie das Programm in DIM-Befehle um. Nicht zu vergessen: das Main-Array – es hat die Aufgabe, beide Koordinaten einer Linie zu speichern.
Setzt man den PLOT-Befehl ab, werden Absolutkoordinaten beider Punkte berechnet und im Feld notiert – also ein Array mit vier Elementen pro Linie.
Beispiel: Die Anweisung DFMAIN ma, 200 erzeugt folgendes Main-Array:

Linie 1	x1	y1	x2	y2
Linie 2	x1	y1	x2	y2
.....				
Linie 199	x1	y1	x2	y2
Linie 200	x1	y1	x2	y2

Außerdem gibt's einen Zeiger (MPTR), der auf die als nächste einzugebende Linie weist. Im Main-Array stehen die Daten jetzt so, wie sie später auf dem Drucker ausgegeben werden. Das geschieht immer nach der EXEC-Anweisung oder wenn ein Feldüberlauf droht (in unserem Beispiel also nach dem 200sten PLOT-Befehl!)

Es gibt noch eine andere Feldvariante im Tolp 64-System: **Sub-Arrays.** Sie übernehmen die Aufgabe, Daten so zu speichern, wie sie bei den PLOT-Befehlen eingetragen werden – also verschiedene Arten von Punkt- und Winkeleingaben. Das Programm merkt sich jetzt die Befehle zur gewünschten Grafik und spielt sie gegebenenfalls (per TRANS-Befehl) wieder ins Main-Array ein. Dem Hauptfeld ist es egal, ob die PLOT-Befehle manuell oder automatisch kommen (also aus dem Sub-Array). Das Demoprogramm »PRO3/ANGABEN« verdeutlicht diese Zusammenhänge.

Maximal sind drei Sub-Arrays möglich. Sollen die PLOT-Befehle nicht aufgezeichnet werden, ist es allerdings nicht nötig, Feld zu definieren. Außer den vier Eingabewerten wird noch ein Statuswert (Art und Form der Eingabe) je Linie gespeichert. Beispiel: DFSUB2,S2,30:



[3] Etwas komplizierter läuft's mit mehreren Vektoren und Graphen – aber es lohnt sich!

Linie 1	x1	y1	x2	y2	status1
Linie 2	x1	y1	x2	y2	status2
....					
Linie 29	x1	y1	x2	y2	status3
Linie 30	x1	y1	x2	y2	status4

Selbstverständlich muß man beachten, daß die Koordinatenwerte im Sub- und Main-Array noch relativ sind: im Sub-Array liegen die tatsächlichen Koordinatenwerte (definiert im PLOT-Befehl des Anwenders). Das Main-Array dagegen enthält die bereits umgerechneten Werte.

Font wechseln

Bei Programmstart ist der Zeichensatz »ger.thin« aktiv. Das Maschinenprogramm hat aber genug Platz (15 Byte), um den Namen eines anderen Default-Fonts zu verwenden. Das geht am besten mit einem Maschinensprache-Monitor (z.B. SMON). Laden Sie Tolp 64, starten es aber nicht und durchsuchen Sie den Speicherbereich von $0801 (2049) bis $3C7E (15486) nach der Zeichenfolge »ger.thin«: ab Adresse $0948 (2376) wird man fündig. In den Bereich bis maximal $0958 (2392) darf man den Namen eines weiteren Fonts (z.B. von unserer Sonderheftdisk) eintragen: ger.double, cbm.thin oder greek – aber vergessen Sie nicht die Anführungszeichen jeweils an Anfang und Ende des Fontnamens! Wer einen der neuen Fonts zum Default-Zeichensatz machen will, muß das geänderte Programm Tolp 64 erneut speichern. Wer keinen Maschinensprache-Monitor besitzt, hat bedeutend mehr Umstände. Tippen Sie zunächst das kurze Basic-Listing ab und speichern es auf Disk (z.B. mit dem File-Namen »fontchange«):

```
10 INPUT "FONTNAME";NA$
20 FOR I=0 TO LEN(NA$)-1
30 POKE 2377+I,ASC(MID$(NA$,I+1,1))
40 NEXT: POKE 2377+I,34
```

Laden Sie jetzt Tolp 64, starten es aber nicht. Geben Sie folgende POKE-Befehle im Direktmodus ein:

```
POKE 44,64: POKE 16384, 0
```

Damit verlegt man den Basic-Anfang nach $4001 (16385). Sie holen jetzt »fontchange« in den Speicher, starten es und tragen den gewünschten Font-Namen ein (mit <RETURN> abschließen!). Wenn sich der Cursor wieder mit READY meldet, ist der Original-Basic-Anfang wieder herzustellen:

```
POKE 44,8
```

Startet man jetzt Tolp 64 mit RUN, beweist die Lademeldung, daß das Programm nun auf den gewünschten Font zurückgreift.

Die neuen Befehle

Hier die ausführliche Übersicht zu den zusätzlichen Basic-Anweisungen:

PIXEL horizontal, vertikal

Zweck: Einstellung der Grafikdichte. Die Parameter: horizontal = 0 bis 7, vertikal = 1 bis 3.

Der Wert für die horizontale Auslösung entspricht den Grafikdruckmodi der Epson-Drucker:

horizontal	Bezeichnung	Dichte (Punkte/Zoll)	Auslösung
0	einfach	60	0,42 mm
1	doppelt	120	0,21 mm
2	doppelt + schnell	120	0,21 mm
3	vierfach	240	0,11 mm
4	CRT-Bildschirm	80	0,31 mm
5	Eins-zu-eins	72	0,35 mm
6	CRT hochauflösend	90	0,28 mm
7	Zwei-zu-zwei	144	0,18 mm

Der Vertikalwert legt den Vorschub je Punktreihe fest:

vertikal	Dichte (Punkte/Zoll)	Auflösung
1	72	0,35 mm
2	144	0,18 mm
3	216	0,12 mm

Rechenaufwand und Zeitfaktor erhöhen sich dramatisch, je besser die Auflösungen sind!

Beispiel: PIXEL 0,1 (Einstellung mit vertretbarem Rechenaufwand).

INIT

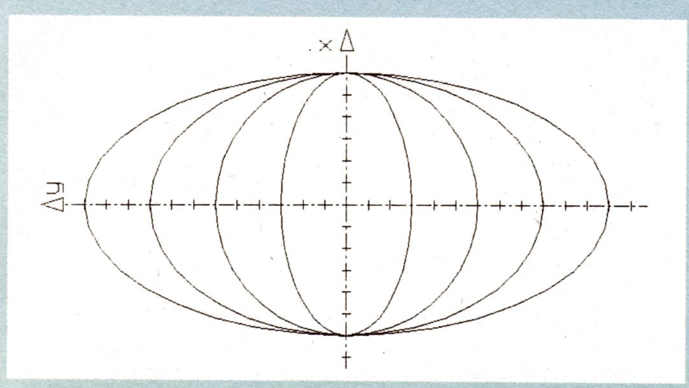
Zweck: Initialisierung der Grafikspeicher und Rückstellung wesentlicher Werte.

INIT muß unbedingt vor dem ersten Zeichenbefehl eingegeben werden, am besten direkt nach DEFINE (DFMINE, DFSUB) oder PIXEL. Außerdem aktiviert die Anweisung die wichtige Interrupt-Routine zum »Anhalten« des Rechners nach <RUN/STOP RESTORE>.

DFMIN var, zahl

Zweck: Dimensionierung des Main-Arrays. Die Parameter: var = beliebiger Name fürs Array, zahl = 1 bis 255

Die angegebene Zahl entspricht der Linienanzahl, die das Main-Array zwischenspeichern soll. Ist der Wert erreicht, startet der Ausdruck automatisch. Nur ein einziges Main-Array ist möglich. Bei Redefinierung erhält man die Fehlermeldung: »Main yet Defined Error«. Beispiel: DFMAIN ma, 50 (Dimensionierung des Main-Arrays »ma« für 50 Linien).



[4] Exakt proportional: Anzeige in x- und y-Richtung

Tolp 64 – Grafikwunder

Drucker als Grafikbildschirm

DFSUB nr, var, zahl
Zweck: Definition eines der drei möglichen SUB-Arrays. Die Parameter: nr = 1 bis 3, var = Array-Name, zahl = 1 bis 255.

Von allen weiteren Befehlen läßt sich jetzt das Array mit Nummer oder per Variablennamen ansprechen. »zahl« ist die Menge der PLOT-Befehle, die das definierte Sub-Array speichern soll. Jedes Sub-Array darf im Programm nur einmal vorkommen – sonst erscheint die Fehlermeldung: »Sub(zahl) yet Defined Error«.

Beispiel: DFSUB 1,rt,30 (Dimensionierung des ersten Sub-Array »rt« für 30 Linien).

STORE zahl
Zweck: Benutzerbereich des Main-Arrays festlegen. Parameter: zahl = 1 bis 255.

Vorteilhaft ist, nicht stets das gesamte Main-Array zur Zwischenspeicherung zu verwenden, sondern eine bestimmte Anzahl Linien (zahl). Achtung: nach jeder DFMAIN-Anweisung ist STORE anzugeben! Der Maximalwert für »zahl« ist mit dem im DFMAIN-Befehl identisch. Wird er überschritten, kommt die Fehlermeldung: »too much lines error«.

Beispiele: DFMAIN ma,40: STORE 30 (das definierte Main-Array enthält maximal 30 Linien. Nach Eingabe des 30. PLOT-Befehls wird die Grafik ausgedruckt.)

STORE 40 (erlaubt dem Anwender, die volle Kapazität von 40 Linien zu nutzen)

MPTR zahl
Zweck: bestimmt die gültig gespeicherte Linie im Main-Array. Parameter: zahl = 0 bis 255.

Der Befehl setzt einen Zeiger, der auf die Linie im Main-Array zeigt, die per PLOT-Anweisung eingegeben wird. Befinden sich z.B. bereits 30 Linien im Main-Array, richtet sich der Zeiger auf die Position der 31sten Linie.

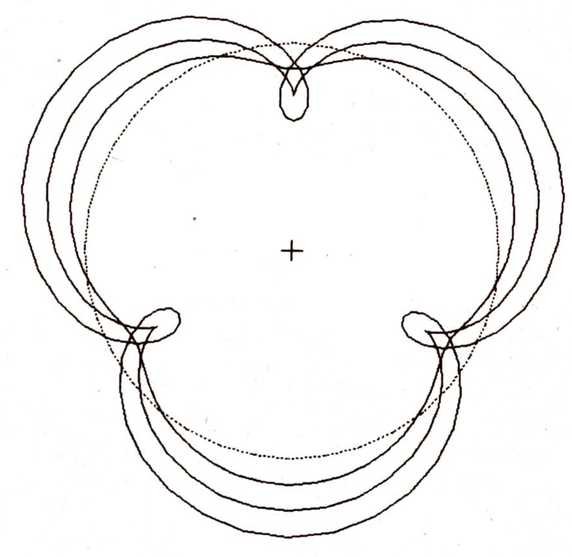
Gibt man dann aber »MPTR 20« ein, macht das Programm die zehn letzten Linien zum Überschreiben frei (z.B. beim nächsten PLOT-Befehl).

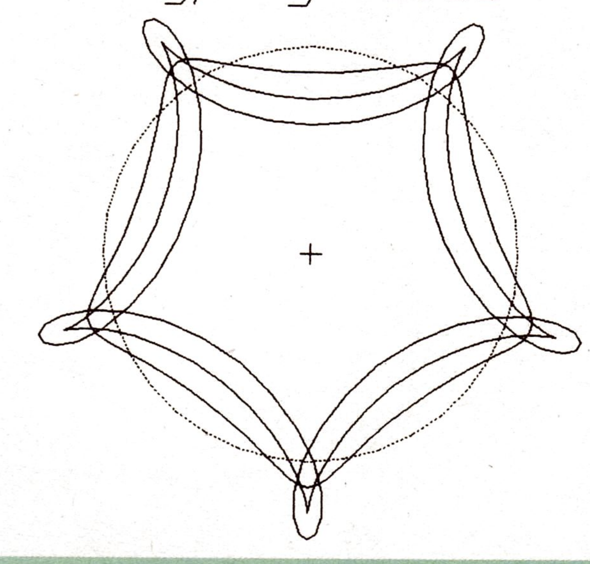
Achtung: Trägt man bei MPTR den gleichen Wert ein wie beim vorhergehenden STORE-Befehl, wird der Grafikdruck aktiviert. Das kann z.B. in folgendem Fall sinnvoll sein:

Ein Koordinatenkreuz soll mit allen Achsen, Skalierungsstrichen und Pfeilen – ab MPTR 0 beginnend – per PLOT gezeichnet werden (die entsprechenden Daten müssen dann ganz vorne im Feld stehen). MPTR 0 bedeutet, daß 0 Linien gelten und das Main-Array bei Linie 1 beginnend ausgefüllt wird.

Für die PLOT-Anweisungen hat man z.B. 23 Linien vorgesehen. Auf zwei Papierblättern möchte man verschiedene Grafiken ausgeben, die aber dasselbe Koordinatenkreuz benötigen.

Nach den Grafikbefehlen wird nun der Wert des MPTR (in diesem Fall 23) in der Basic-Variablen »pt« zwischengespeichert, die erste Grafik durch PLOT-Befehle erzeugt und mit EXEC ausgedruckt. Nach dem Papierwechsel muß das Programm die Anweisung »MPTR pt« und die Daten zur





[5] Auch grafische Muster verschiedener Art auf mathematischer Basis sind prolbemlos möglich

PIXEL 3, 3
PIXEL 7, 3
PIXEL 7, 2
PIXEL 5, 1
PIXEL 1, 2
PIXEL 0, 1
PIXEL 6, 1

[6] Einzelbuchstaben wurden als Vektoren aufgebaut

zweiten Grafik erhalten, die wiederum per EXEC gedruckt wird. Diese effektive Anwendung des MPTR-Befehls findet man im Demoprogramm »pro1/mptr«.

Die Anwendung nützt nur bei Grafiken etwas, die aus weniger als oder maximal 255 Linien bestehen (Einfachgrafiken).

SPTR sub, zahl

Zweck: Bestimmen der gültig gespeicherten PLOT-Befehle im entsprechenden Sub-Array. Parameter: sub = Sub-Array-Nummer, zahl = 0 bis 255.

Die Anweisung wirkt analog zu MPTR und wird weitaus häufiger verwendet. Da es mehrere Sub-Arrays gibt, muß man stets das jeweilige Feld angeben. Das geht entweder mit der Feldbezeichnung SPTR;s1,20 (Array-Nummer identisch mit dem Wert in DFSUB) oder SPTR1,20. Das Ergebnis ist stets dasselbe: in Sub-Array 1 werden 20 gültige PLOT-Befehle gesetzt.

Beachten Sie das Programmbeispiel »pro2/sptr« auf der Diskette zu diesem Sonderheft.

SHSUB sub

Zweck: Ausgabe aller gültig gespeicherten PLOT-Befehle eines Sub-Arrays. Parameter: sub = Sub-Array-Nummer.

Das jeweilige Sub-Array wird hier ebenfalls entweder über die Feldbezeichnung oder über die Feldnummer angegeben. Wurde z.B.DFSUB3,tr,40 definiert, liefern SHSUB3 und SH-SUB,tr dasselbe Ergebnis.

Durch Verstellen des jeweiligen Array-Pointers (SPTR) läßt sich die Anzahl der gültigen Linien verändern.

ACT sub

Zweck: Aktivierung oder Deaktivierung des aktiven Sub-Arrays. Parameter: sub = Sub-Array-Nummer (ein) oder 0 (aus).

... bestimmt das Sub-Array, das die eingegebenen PLOT-Befehle speichert. Sollen die Befehle nicht protokolliert werden, ist ACT 0 am Zug. Dann würden die PLOT-Befehle nur im Main-Array Veränderungen einleiten.

Läuft das aktuelle Sub-Array über, führt das Programm die Anweisung ACT 0 automatisch aus. Auf die Tatsache, daß ab sofort keine PLOT-Befehle mehr protokolliert werden (das geschieht erst wieder, wenn man ein anderes Sub-Array aktiviert), wird der User vom Programm automatisch aufmerksam gemacht:

Bei gesetztem MESS-Modul (s. MESS) gibt der Befehl eine Warnung aus und ändert, wie auch bei deaktiviertem MESS, den Inhalt der Res-Adresse. Nach PLOT ohne Überlauf des aktiven Sub-Arrays steht in der Res-Adresse der Wert 0; beim Überlauf wird er auf <> 0 gesetzt.

SHSYS ['[nk]]

Zweck: zeigt die im Speicher vorhandenen Koordinatensysteme. Parameter: nk = 1 bis 4.

»nk« ist die Anzahl der Nachkommastellen. Sollen die Winkel in Altgraden erscheinen, muß man den Apostroph zusätzlich angeben. Fehlt die Nachkommastellenzahl, übernimmt das Programm den Wert der letzten Anzeige. Nach Programmstart ist »2« voreingestellt.

Beschreibung des Bildschirm-Displays:
- * = aktives Relativsystem
- 1 bis 3 sind die (mit COPY) abrufbaren Relativsysteme, die derzeit nicht aktiv sind. Zu jedem System sind die Parameter angegeben, der Koordinatenursprung als Absolutwerte.
- Winkel zwischen relativer und absoluter x-Achse
- Winkel zwischen relativer x- und y-Achse
- Skalierungswert für die relative x- und y-Achse
- Teilungsart der relativen Achsen (linear und logarithmisch)
- Skalierungswert für die absolute x- und y-Achse (bei allen Relativsystemen gleich!)

AXIS punkt, winkel [, winkel [, ab]], <llog>

Zweck: Setzen des Koordinatenursprungs, des Winkels zwischen relativer und absoluter x-Achse, des Winkels zwischen den beiden relativen Achsen und der Teilung der Relativachsen

»punkt« ist der Koordinatenursprung des Relativsystems, der sofort in Absolutwerte umgerechnet und gespeichert wird. Der erste (unbedingt anzugebende) Parameter »winkel« kann jede Zahl zwischen relativer und absoluter x-Achse sein, der zweite (optionale) darf zwischen relativer x- und relativer y-Achse liegen. Diesen Wert muß man nur dann angeben, wenn er ungleich 90 Altgrad ist. Die Variablen »a« und »b« bezeichnen die Teilungsarten der Relativachsen:

– a: Teilung der x-Achse,
– b: ... der y-Achse.

»l« bedeutet lineare, »log« logarithmische Teilung.

Beispiele:
– LLOG x-Achse ist linear, y-Achse ist logarithmisch geteilt,
– LOGLOG: beide Achsen logarithmisch geteilt,
– LL: bilaterale lineare Teilung ist optional.

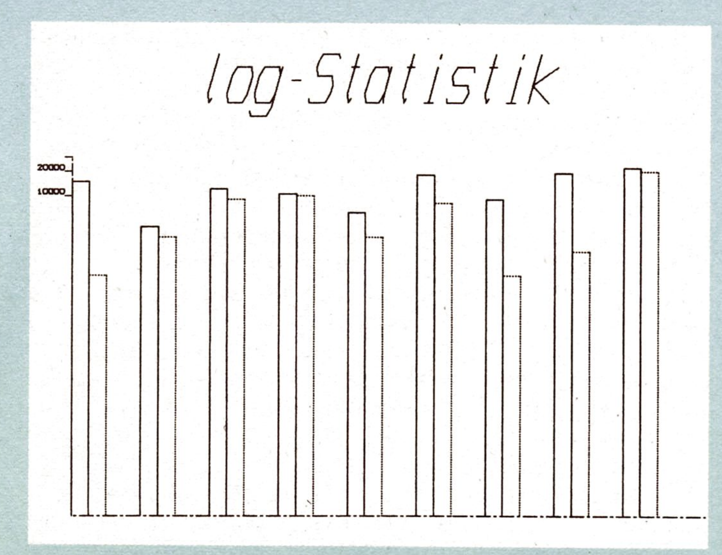
SCALE sx, sy

Zweck: Relativachsen-Skalierung festlegen. Parameter: sx, sy = beliebige Werte.

Beispiel: SCALE 2,1 dehnt die x-Achse auf doppelte Länge. Bei Angabe negativer Werte lassen sich Grafiken spiegeln.

PAPER px, py

Zweck: Absolutachsen-Skalierung festlegen. Parameter:



[7] Grafische Auswertung mit Tolp: Statistik-Beispiel

px, py = beliebige Werte für x-uy-Achse.

Auch hier kann man Grafiken spiegeln, wenn man negative Werte einträgt.

COPY a, b

Zweck: Koordinatensysteme verschieben. Parameter: a, b = 0 bis 3

Um augenblicklich ruhende Koordinatensysteme (1, 2, 3) zu aktivieren, muß man das gewünschte System in den Speicher des aktuell gültigen (Kennziffer 0) kopieren. System »a« wird stets in den Bereich von System »b« übertragen.

Beispiele: COPY 3,0 (aktiviert System 3)

COPY 0,2 (überträgt das aktive System in Koordinatenspeicher 2)

SVSYS open, file

Zweck: Koordinatensysteme speichern. Parameter: open = übliche OPEN-Anweisung an die Floppy (z.B. OPEN 2,8,2), file = Dateiname

... sichert das aktive Koordinatensystem und die drei deaktivierten Systeme auf Diskette. Achtung: Solche Systemdateien werden im Directory nicht speziell gekennzeichnet.

Beispiel: SVSYS 2,8,2,"Trochoide"

LDSYS open, file

Zweck: Koordinatensysteme laden: Parameter: open, file = s. SVSYS

.. holen vorher per SVSYS gespeicherte Koordinatensysteme wieder in den Computer.

Beispiel: LDSYS2,8,3,"coord-sys 1"

Skalierungswerte des Absolutsystems werden nicht gesichert.

Zeichenbefehle

Tolp 64 kennt viele Möglichkeiten, Grafikpunkte zu bestimmen. Zuvor einige Bemerkungen zu den Koordinatensystemen.

Beim **Absolutsystem** liegt die x-Achse in der linken Kante des DIN-A4-Blatts, die y-Achse in der oberen Kante. Hier kann man lediglich die Skalierung der Achsen verändern (s. PAPER).

Beim **Relativsystem** lassen sich Koordinatenursprung, Winkel der relativen zur absoluten x-Achse, Winkel zwischen relativer x-Achse und relativer y-Achse, Skalierung der beiden Achsen sowie Teilung (linear oder logarithmisch) einstellen. Alle Längenangaben der Zeichenbefehle beziehen sich auf Zentimeter (cm).

Nun zu den möglichen Punkt- und Winkelangaben (s. Demo »pro3/angaben«):

Tolp 64 – Grafikwunder

Drucker als Grafikbildschirm

```
!"äö%ü'()Ä+,-./0123456789:;ö=Ü
?βαβγδεζηθικλμνξοπρστυφχψωyz[
]ΑΒΓΔΕΖΗΘΙΚΛΜΝΞΟΠΡΣΤΥΦΧΨΩΥΖ
!"#$%&'()*+,-./0123456789:;<=>
?@abcdefghijklmnopqrstuvwxyz[
£]ABCDEFGHIJKLMNOPQRSTUVWXYZ
!"äö%ü'()Ä+,-./0123456789:;ö=Ü
?ßabcdefghijklmnopqrstuvwxyz[
]ABCDEFGHIJKLMNOPQRSTUVWXYZ
!"äö%ü'()Ä+,-./0123456789:;ö=Ü
?ßabcdefghijklmnopqrstuvwxyz[
]ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

[8] Zeichensätze lassen sich nach Wunsch definieren

Winkelangaben

[x0]30'

... bedeutet, daß sich der Winkel auf eine der x-Achsen bezieht (bei »y0« wäre es eine der y-Achsen) Die Zahl 0 hinter dem Buchstaben markiert, daß es sich um die absolute Achse handelt. Der Winkel beträgt 30 Altgrad.

[y1]2.35

Hier bezieht sich der Winkel auf die relative y-Achse, da hinter dem Buchstaben (y) die »1« steht. Wurde im ersten Beispiel der Winkel noch in Altgrad vermerkt (erkennbar am Apostroph), interpretiert unser zweites Beispiel die Winkelangabe »2.35« als Radiant.

[x1]rt'

Der in der Basic-Variablen »rt« gespeicherte Wert (Altgradwinkel) weist auf die relative x-Achse hin. Intern rechnet das Programm alle Winkel auf die absolute x-Achse um (wichtig bei der SHSY-Anweisung!).

Bei der Winkelangabe kann man die eckige Klammer auch weglassen: das Programm nimmt dann automatisch »[x1]« an. Praktische Anwendungsbeispiele: s. Befehlserläuterung zu PLOT.

Punktangaben

[k0]2,3

Die Anweisung »k« bedeutet, daß die beiden angegebenen Zahlen als karthesische Koordinaten zu interpretieren sind. Da die Zahl 0 folgt, beziehen sich die x/y-Angaben aufs Absolutsystem.

[k1]1,2

... und hier aufs aktuelle Relativsystem. Die Angabe von [k1] ist optional, »1, 2« allein würde dasselbe Ergebnis bringen. Außer »0« und »1« ist bei Punktangaben auch noch die Ziffer 2 möglich. Dann bezieht sich die Punktangabe auf die

Druckerhinweise

Die Druckroutinen von »Tolp 64« wurden für den Epson-kompatiblen Citizen 120 D entworfen. Sie funktionieren auch mit anderen Geräten, die seriell mit dem C 64 verbunden sind, wenn man die Sekundäradresse (= Linearkanal für Grafikdruck) anpaßt (z.B. »1« für den Star-LC 10, Epson-FX 80/85 usw.). Der entsprechende Wert steht in Speicherstelle $201A (8218). Die Programmversion von Tolp 64 auf Disk ist auf Sekundäradresse 1 voreingestellt. Sollte Ihr Gerät einen anderen Wert brauchen, müssen Sie ihn nach dem Laden von Tolp 64, aber vor dem Start mit RUN, in der entsprechenden Speicherstelle eintragen:

```
POKE 8218, zahl
```

Eventuell müssen Sie auch den DIP-Schalter für den Zeilenvorschub (LF) korrigieren (on/off).

aktuelle Position des virtuellen (= angenommenen) Plotterstiftes. Gibt man z.B. die Anweisung »PLOTTO [k0] 1,2«, steht der unsichtbare Plotterstift auf Position 1,2 (k0 = Absolutsystem).

Weiteres Beispiel:

```
PLOT[K2] 2,-1 TO [K0]3,4
```

Jetzt findet man den Anfangspunkt der Linie bei den Koordinaten 3,1 (1 + 2 = 3 und 2 + (-1) =1) wieder.

Alle Angaben zu karthesischen Koordinaten gelten auch für polare.

[p0] 1.4,[y0] 30'

... bezeichnet einen Punkt, der 1,4 Einheiten vom Absolutkoordinatenursprung entfernt liegt und dessen Verbindung mit dem Ursprung und der absoluten y-Achse einen Winkel von 30 Altgrad einschließt.

Polarkoordinaten kann man natürlich auch in Relativsystemen ([p1]) oder für die Stellung des virtuellen Plotterstifts ([p2]) verwenden.

Es gibt noch einen weiteren Modus, Punkte zu definieren: **Ketteneingabe** (s. Beispielprogramm »pro4/kettenang.«). Damit kann man interne Vektoradditionen durchführen. Sinnvoll ist diese Funktion vor allem bei Polarkoordinaten.

```
PLOT [P1] 5,GA'&[P1] 3,2*GA'& [P1]
1,3*GA'TO0,0
```

Diese Befehlszeile leitet für den ersten Punkt intern folgende Berechnung ein:

(5)	(3)	(1)	(r)
( ) +	( ) +	( ) =	( )
(ga')	(2*ga')	(3*ga')	(alpha)

Da die drei Vektoren an Schaft und Spitze stets zusammenhängen, aber unterschiedliche Rotationsgeschwindigkeiten um den jeweils vorhergehenden Vektor haben, lassen sich damit komfortabel komplexe Aufrad- und Inradlinien zeichnen. Kleine Einschränkung: verkettete PLOT-Befehle kann man nicht im aktiven Sub-Array notieren, außerdem muß man zuvor unbedingt ACT 0 aktivieren!

PLOT punkt TO punkt
PLOT TO punkt
PLOT punkt TO punkt; m
PLOT TO punkt; m

Zweck: Zeichnen einer Linie zwischen zwei Punkten. Parameter: punkt = x- und y-Koordinate, m = 0 bis 255

»m« bestimmt die Linienart (s. MODE). Der so eingestellte Typ gilt beim PLOT-Befehl jeweils nur für die aktuelle Linie.

MOVE punkt

Zweck: Setzen des virtuellen Plotterstifts. Parameter: punkt = x- u y-Koordinatenpaar

... legt den Ausgangspunkt der PLOT TO-Anweisung fest. Beispiele:

```
MOVE [k0] 4,5:PLOTTO [p1] 3, [y1] 15'
```

ist äquivalent zu

```
PLOT [k0] 4,5to [p1] 3, [y1] 15'
```

MODE modus

Zweck: Linienart einstellen. Parameter: modus = 0 bis 255 (0 = Vollinie, 1 = punktierte Linie).

Was die übrigen Werte bedeuten, zeigt Ihnen der Screen nach der Anweisung SHMODE. Gibt man beim PLOT-Befehl keine Linienart an, verwendet das Programm den aktuellen Parameter »modus« von MODE.

SHMODE modus [;]

Zweck: Anzeigen des Linienmusters. Parameter: modus = 0 bis 255.

Damit können Sie 256 verschiedene Linienarten jeweils in der obersten Bildschirmzeile betrachten. Folgt der Anweisung ein Semikolon, wird der Wert sofort als aktuelle Linienart übernommen. Beispiel:

SHMODE2; (äquivalent zu SHMODE2: MODE2).

BOUND punkt TO punkt

Zweck: Rechteck auf dem Papier festlegen

Der erste Punkt (also die x/y-Koordinaten) definiert die linke obere, der zweite die rechte untere Ecke des Quaders, das auf dem Papier den Zeichenbereich umgrenzt. Damit klammert man automatisch Bereiche aus, in denen keine Grafik erscheint – das vermindert die Rechenzeit erheblich!

TRANS sub

Zweck: Übertragen mehrerer PLOT-Befehle ins Main-Array. Parameter: sub = Sub-Array-Nummer

Hier sind wieder die Formen TRANS,s1 (Namen des Sub-Arrays) oder TRANS1 (Nummer) möglich – selbstverständlich abhängig vom Eintrag bei DFSUB.

Dieser Befehl ist mächtig: Protokolliert man die PLOT-Befehle einem Sub-Array mit, kann man zu einem späteren Zeitpunkt (nach dem Ausdruck der ersten Grafik und Zurücksetzen des MPTR auf 0) dieselbe Grafik in ein anderes Koordinatensystem einfließen lassen.

Wichtig sind hier die SPTR-Werte des entsprechenden Sub-Arrays: sie legen fest, bis zu welchem PLOT-Befehl das Sub-Array bearbeitet werden soll.

```
SPTR3,10:TRANS3
```

würden z.B. die Ergebnisse der ersten zehn PLOT-Befehle des Sub-Arrays 3 im Main-Array ablegen.

LDFONT string$

Zweck: lädt Zeichensatz für die WRITE-Anweisung. Parameter: string$ = Font-Name.

Fonts werden im Inhaltsverzeichnis der Arbeitsdiskette nicht speziell gekennzeichnet (normaler Dateityp PRG). Mit dem Zusatzprogramm »EDITOR.TOLP« (s. Beschreibung) kann man komfortabel neue Zeichensätze erzeugen.

WRITE punkt; string$, b, h
WRITE; string$, b, h
WRITE punkt; string$, b, h; winkel, winkel
WRITE; string$, b, h; winkel, winkel

Zweck: Textausgabe

»punkt« ist die Koordinate des linken unteren Pixels des ersten Buchstabens. Fehlt dieser Parameter, gilt die augenblickliche Position des virtuellen Plotterstifts. Nach WRITE steht er direkt hinter dem letzten Zeichen. Man kann also ohne weitere Koordinatenangabe Text anhängen. Die Zeichenkette darf eine Konstante oder eine Basic-Variable sein. Enthält der String Zeichen, die nicht im Font enthalten sind (z.B. Grafikzeichen), provoziert man die Fehlermeldung: Illegal Character Error. Wesentlich ist, daß WRITE-Befehle nicht im Sub-Array erscheinen, sondern lediglich im Main-Array. Kommt's zu einem Überlauf, wird der Druck automatisch eingeleitet.

Zu jedem WRITE-Befehl muß man zusätzlich Breite (Parameter: b) und Höhe (h) des Buchstabens angeben. Für die beiden Längenmaße gibt's wieder verschiedene Möglichkeiten der Parameterangabe, z.B.:

```
WRITE;"Text", 0 2,1 4
```

»0« Länge bedeutet, daß der Wert 2 nur der Absolutskalierung gehorcht (s. PAPER).

Die Basic-Variablen in »header.tolp«

as	2Byte	Name des aktuellen Sub-Arrays
an	1Byte	Nummer des aktuellen Sub-Arrays
mo	255Byte	Feld aller Linienarten für jede Linie (z.B.: 5. Linie (MEM(mo+5))
rl	Fließkommaw.	(REL-Wert+1)/15
ma	2Byte	Namen des Main-Array
s1	2Byte	Namen des Sub-Arrays 1
s2	2Byte	Namen des Sub-Arrays 2
s3	2Byte	Namen des Sub-Arrays 3
px	Fließkommaw.	Skalenwert der absoluten x-Achse
py	Fließkommaw.	Skalenwert der absoluten y-Achse
sx	Fließkommaw.	Skalenwert der RELativen x-Achse
sy	Fließkommaw.	Skalenwert der RELativen y-Achse
gx	Fließkommaw.	x-Koordinate des virtuellen Plotterstiftes
gy	Fließkommaw.	y-Koordinate des virtuellen Plotterstiftes
nk	1Byte	Nachkommastellen beim SHSYS-Befehl
so	1Byte	STORE-Wert
mp	1Byte	MPTR
p1	1Byte	SPTR von SUB1
p2	1Byte	SPTR von SUB2
p3	1Byte	SPTR von SUB3
lx	Fließkommaw.	x-Wert des linken oberen Ecks des BOUND-Fensters
ly	Fließkommaw.	y-Wert des linken oberen Ecks des BOUND-Fensters
rx	Fließkommaw.	x-Wert des rechten unteren Ecks des BOUND-Fensters
ry	Fließkommaw.	y-Wert des rechten unteren Ecks des BOUND-Fensters
l0		Vollinie
l1		punktiert
l2		kurzstrichliert
l3		langstrichliert
l4		kurzstrichpunktiert
l5		langstrichpunktiert
l6		strichpunktiert mit drei Punkten

Die Höhe wird zusätzlich durch die Relativskalierung (s. SCALE) beeinflußt und erscheint dadurch gestreckt. Die Breite ist gegen solche Verzerrungen gefeit. Mit negativen Werten für b und h lassen sich interessante Effekte (Spiegelschrift) erzielen.

Das Verhältnis der Breite jedes Zeichens zum Abstand zwischen zwei Zeichen läßt sich nicht durch WRITE bestimmen – das besorgt der nächste Befehl:

REL fac

Zweck: Einstellung des Abstands zwischen zwei Zeichen. Parameter: fac = Fließkommawert, ungleich »0«.

... stellt ein, wie weit die Zeichen voneinander entfernt sind. Näheres s. WRITE.

EXEC

Zweck: Druckausgabe der im Main-Array gespeicherten Linien

Enthält das Hauptfeld keine Linien, wird (je nach Meßmodus) eine Warnung ausgegeben.

Es erscheinen immer so viele Linien auf dem Papier, wie unter MPTR angegeben. Wenn der EXEC-Befehl seine Arbeit beendet hat, erhält MPTR stets den Wert 0.

Erweiterte Basic-Befehle

JUMP zeile

Zweck: GOTO-Befehl für berechnete Sprungziele. Parameter: zeile = im erlaubten Nummernbereich

Beispiel: A = 34:JUMP A

DSTAT

Zweck: Ausgabe des Disketten-Fehlerstatus

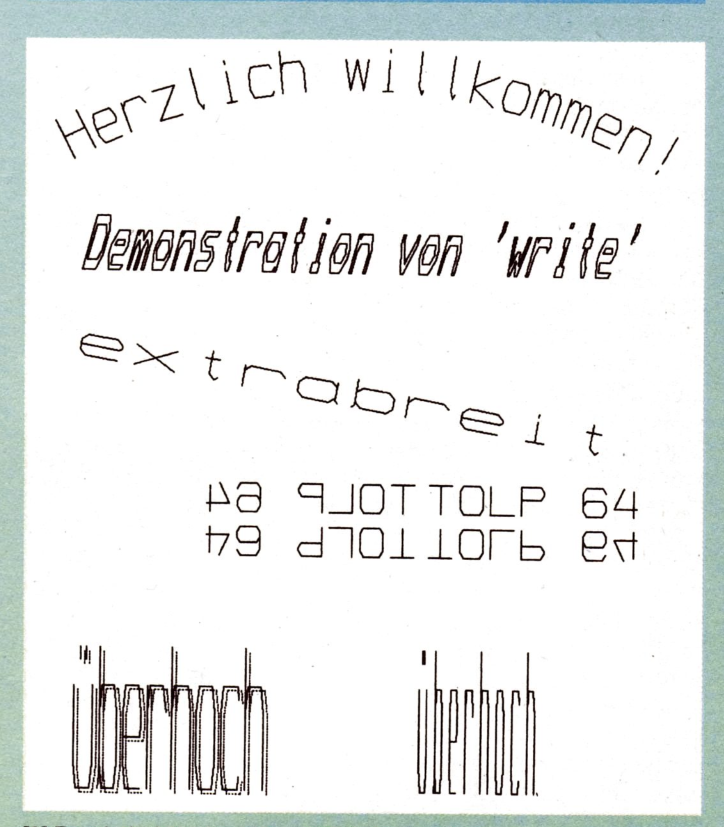
... berücksichtigt stets Laufwerk Nr. 8.

CLA

Zweck: löscht Main- und Sub-Arrays und führt den CLR-Befehl aus.

Tolp 64 – Grafikwunder

Drucker als Grafikbildschirm



[9] Fast jede Schriftart ist mit Tolp 64 machbar

Vermeiden Sie die CLR-Anweisung des Basic 2.0 und verwenden Sie stets CLA, um Kompatibilitätsproblemen aus dem Weg zu gehen.

RR

Zweck: Starten des Basic-Programms

Der Befehl entspricht RUN, allerdings aktiviert er vorher den Befehl CLA.

DIR

Zweck: Anzeige des Inhaltsverzeichnisses der aktuellen Disk in Laufwerk Nr. 8

RES adr

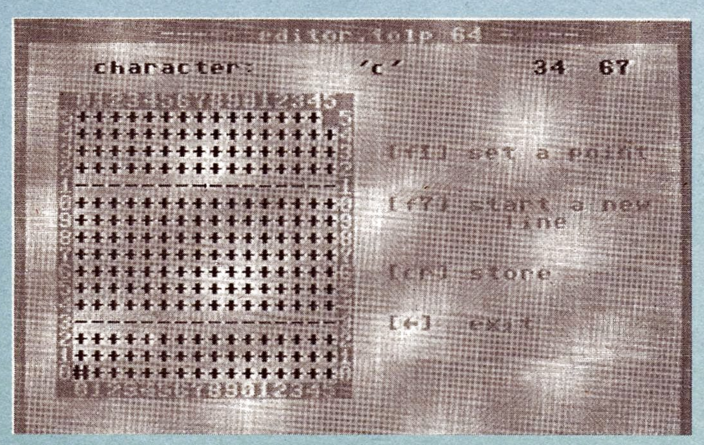
Zweck: Result-Adresse festlegen. Parameter: adr = Speicherstelle.

Die Funktionen ACT und MEM liefern einen Wert in einer vom Anwender vorbestimmten Adresse. Voreingestellt ist Speicherzelle 830 (s. MEM und ACT).

MEM adr [;]

Zweck: liest den RAM-Inhalt einer Adresse. Parameter: adr = Speicheradresse

Es wird stets der RAM-Inhalt herausgefiltert. Der Unter-



[10] Editorbildschirm für eigene Zeichensätze

schied zwischen den beiden Syntax-Formen:

`MEM 40960`

... ohne Semikolon gibt den Inhalt von Adresse 40960 auf dem Bildschirm aus und notiert den Wert in der RES-Adresse; mit Strichpunkt entfällt die Screen-Ausgabe.

In Verbindung mit den Basic-Variablen (definiert im File »header.tolp«) lassen sich wichtige Statusinformationen holen. Beispiel: MEM mp liefert den Wert des MPTR (Voraussetzung: mp muß in header.tolp definiert sein.)

FLT adr, var

Zweck: Einlesen eines Fließkommawertes in eine Basic-Variable. Parameter : adr = Speicheradresse, var = Basic-Variable

Die Speicherzelle »adr« ist die Adresse des ersten Bytes einer 5-Byte-Fließkommafolge. Auch hier enthält header.tolp nützliche Werte.

Beispiel: FLT PX, A überträgt den x-Skalenwert fürs Absolutsystem (s. PAPER) in die Basic-Variable A .

FFEED

Zweck: Seitenvorschub am Drucker mit der Geräteadesse 4

COLOR hf, rf, s1, s2 [;]

Zweck: Setzen der Bildschirmfarben: Parameter: hf = Hintergrundfarbe, rf = Rahmenfarbe, s1 = Schriftfarbe 1, s2 = Schriftfarbe 2

In der ersten Syntax-Form (ohne Semikolon) werden die angegebenen Farben nur zwischengespeichert, aber nicht gesetzt. Dazu braucht man den SHSYS-Befehl. Mit Strichpunkt wählt man dagegen den direkten Weg (ohne SHSYS).

»1« ist die normale Bildschirmschriftfarbe, »2« dagegen hat nur beim SHSYS-Befehl eine Bedeutung.

MESS wert

Zweck: Ein- bzw. Ausschalten des Message-Modus. Parameter: wert = 0 (aus), 1 (an).

... stellt, ob zu bestimmten Befehlen Warnungen auf dem Bildschirm erscheinen:

– EXEC: Meldung, daß im Main-Array keine Linie gespeichert ist.

– PLOT: Warnung, daß im aktuellen Sub-Array ein Überlauf aufgetreten und kein Sub-Array aktiv ist.

Die Abbildungen 2 bis 7 zeigen eine Fülle grafischer Muster, die mit den Zeichenbefehlen von Tolp 64 erzeugt wurden.

Die Initialisierungsdatei

In header.tolp sind Basic-Variablen mit wesentlichen Adressen gespeichert, die bei Programmstart initialisiert werden. Diese Adressen geben mit MEM und FLT wichtige Infos an den Programmierer weiter. Da es sinnvoll ist, diese Datei (oder Teile daraus) als Grundlage für eigene Tolp-Programme zu verwenden, erscheint nach der Einschaltmeldung der entsprechende Ladebefehl. Außer dem Titel, der sich in header.tolp verewigen läßt, sollte man auch den Grafiktyp (Plotter- oder Einzelgrafik) im Programmkopf festhalten.

Zeichensätze definieren

Mit dem zusätzlichen Basic-Programm »editor.tolp« auf Diskette, kann man sich eigene Fonts zusammenstellen (Abb. 8 und 9).

Jeder Buchstabe besteht aus einer 15 x 15-Punkte-Matrix, wobei sich aber der rechte obere Punkt nicht nutzen läßt. Wurden bei WRITE Breite und Höhe mit »1« definiert (ebenso die weiteren Skalierungsbefehle), entsprechen 15 Pixel exakt einem Zentimeter.

Laden Sie das Programm (Tolp 64 darf dabei nicht aktiviert sein!) mit:

`LOAD "EDITOR.TOLP",8`

Nach dem Start mit RUN erscheint das Hauptmenü. Die jeweiligen Programmfunktionen aktiviert man per Zahlentasten:

<1> Design a Character

Geben Sie auf der Tastatur das Zeichen ein, das geändert werden soll – entweder das Zeichen selbst oder den ASCII-Code. Drückt man »Pfeil nach links«, erscheint eine Liste aller veränderbarer Zeichen. Dann baut sich der Editorbildschirm auf (Abb. 10). Im linken Feld sind außer dem Raster noch zwei Hilfslinien (normalerweise für die Lage der Kleinbuchstaben) und der Cursor in Rautenform zu erkennen, der sich wie gewohnt mit den Cursor-Tasten steuern läßt. Um einen Punkt zu setzen, drückt man <F1>. Ist das der zweite einer Linie, wird er mit dem ersten verbunden und dient als erster Punkt einer neuen Linie. <F7> beendet den Streckenzug; <F1> legt den ersten neuen Linienpunkt fest.

Unter dem Editierfenster taucht die Anzahl der bereits definierten Daten auf – 30 Punkte sind maximal möglich. »Pfeil nach links« bricht das Editieren ab. Überschreitet man die Anzahl von 30 Bildpunkten, reagiert das Programm so, als hätte man <RETURN> gedrückt – das Zeichen wird in aktueller Gestalt vom Computer übernommen. Dann erscheint die Datenliste des Feldes (32 Byte). Fürs nächste Zeichen baut sich das Editierfenster automatisch auf.

<2> Look at a Character

... zeigt, wie das gewünschte bzw. geänderte Zeichen aussieht. Achtung: vorher muß man selbstverständlich einen Font laden! Per <F1> und <F7> blättert man im Zeichensatz vor oder zurück, <RETURN> zeigt die Daten des jeweiligen Zeichens. »Pfeil nach links« führt ins Hauptmenü zurück.

<3> Loading a Font

Geben Sie den gewünschten Dateinamen an (auf der Sonderheftdiskette sind das z.B. die Fonts ger.thin, ger.double, cbm.thin und greek). Wer eigene Fonts entwerfen möchte, sollte zunächst einen bestehenden laden, diesen ändern und unter neuem Namen speichern.

<4> Saving a Font

... speichert den aktuellen Zeichensatz im Rechner unter beliebigem Fontnamen auf der Disk im Laufwerk 8.

(bl)

Kurzinfo: Tolp 64

Programmart: Grafik/Drucker-Anwendung
Laden: LOAD "TOLP 64",8
Starten: nach dem Laden RUN eingeben
Besonderheiten: separater Zeichensatz-Editor, Grafikbefehle wirken unmittelbar auf den Drucker
Benötigte Blocks: 74 (inkl. Editor)
Programmautor: Thomas Schlögl

So finden Sie die Programme auf der Diskette

DISKETTE SEITE 1

Blöcke	Name	Typ	Seite
Ø	[illegible]		
21	"disklader"	prg	S. 21
Ø	"----------------"	usr	
Ø	"-----grafik-----"	usr	
Ø	"--erweiterung---"	usr	
Ø	"----------------"	usr	
3	"lader alan v7.2"	prg	S. 4
16	"alan v7.2"	prg	
12	"alan ii"	prg	
1	"demo1"	prg	
2	"demo2"	prg	
2	"demo3"	prg	
4	"demo4"	prg	
7	"demo5"	prg	
Ø	"----------------"	usr	
53	"tolp 64"	prg	S. 10
2	"header.tolp"	prg	
11	"ger.thin"	prg	
11	"ger.double"	prg	
11	"cbm.thin"	prg	
11	"greek"	prg	
19	"editor.tolp"	prg	
6	"pro1/mptr"	prg	
4	"pro2/sptr"	prg	
6	"pro3/angaben"	prg	
8	"pro4/kettenang."	prg	
3	"pro5/trans"	prg	
7	"pro6/write"	prg	
2	"pro7/font-show"	prg	
5	"pro8/log-graphik"	prg	
4	"pro9/dichte"	prg	
8	"pro1Ø/titel"	prg	
Ø	"----------------"	usr	
11	"turtle-grafik"	prg	S. 34
2	"turtle-demo 1"	prg	
1	"turtle-demo 2"	prg	
2	"turtle-demo 3"	prg	
3	"turtle-demo 4"	prg	
2	"turtle-demo 5"	prg	
3	"turtle-demo 6"	prg	
2	"turtle-demo 7"	prg	
3	"turtle-demo 8"	prg	
2	"turtle-demo 9"	prg	
18	"turtle-schrift"	prg	
Ø	"----------------"	usr	
6	"3d-funktion v1.2"	prg	S. 39
32	"bildØ"	prg	
32	"bild1"	prg	
32	"bild2"	prg	
Ø	"----------------"	usr	
Ø	"--grafik-tools--"	usr	
Ø	"----------------"	usr	
11	"editor 2x2"	prg	S. 46
4	"player 2x2"	prg	
7	"fast little demo"	prg	
67	"s.example"	prg	
Ø	"----------------"	usr	
89	"char-wandler v3"	prg	S. 47
Ø	"----------------"	usr	
19	"transfile"	prg	S. 44
9	"f.bootfilter"	prg	
9	"f.c64ascii"	prg	
9	"f.startool"	prg	
2	"f.mastertext"	prg	
3	"demo basic prg"	prg	
6	"snake.t"	prg	
4	"demo basic.t"	prg	
2	"startool.quell"	prg	
Ø	"----------------"	usr	
Ø	"--tips & tricks-"	usr	
Ø	"----------------"	usr	
3	"setpixel"	prg	S. 22
4	"draw"	prg	
5	"circle"	prg	
7	"box"	prg	
8	"stadion"	prg	
3	"speedy hires"	prg	
Ø	"----------------"	usr	
Ø	"----diskette----"	usr	
Ø	"---beidseitig---"	usr	
Ø	"----bespielt----"	usr	
Ø	"----------------"	usr	

DISKETTE SEITE 2

Blöcke	Name	Typ	Seite
Ø	[illegible]		
Ø	"----------------"	usr	
Ø	"--grafik-tools--"	usr	
Ø	"----------------"	usr	
95	"sprite edit (dd)"	prg	S. 48
Ø	"----------------"	usr	
25	"big-pic v5"	prg	S. 40
Ø	"----------------"	usr	
1	"loader"	prg	S. 41
14	"sprites"	prg	
8	"grafiktester"	prg	
9	"demofont"	prg	
4	"demografik"	prg	
Ø	"----------------"	usr	
3	"1. game"	prg	S. 42
3	"2. ram editor"	prg	
3	"3. col.+spr.ed."	prg	
1	"4. char.editor"	prg	
88	"game-demo1"	prg	
5	"game-demo2"	prg	
Ø	"----------------"	usr	
Ø	"--picture-show--"	usr	
Ø	"----------------"	usr	
3	"koala-show 2.2"	prg	S. 49
3	"show.obj $5ØØØ"	prg	
4Ø	"▒pic 1 "	prg	
4Ø	"▒pic 2 "	prg	
4Ø	"▒pic 3 "	prg	
4Ø	"▒pic 4 "	prg	
4Ø	"▒pic 5 "	prg	
4Ø	"▒pic 6 "	prg	
4Ø	"▒pic a london-u"	prg	
Ø	"----------------"	usr	
5	"fundisplay $1ØØØ"	prg	S. 50
65	"↑↑ ferrari 365"	prg	
45	"↑↑ sweet heart"	prg	
Ø	"----------------"	usr	
Ø	"-------ende------"	usr	
Ø	"----------------"	usr	

WICHTIGE HINWEISE

zur beiligenden Diskette:

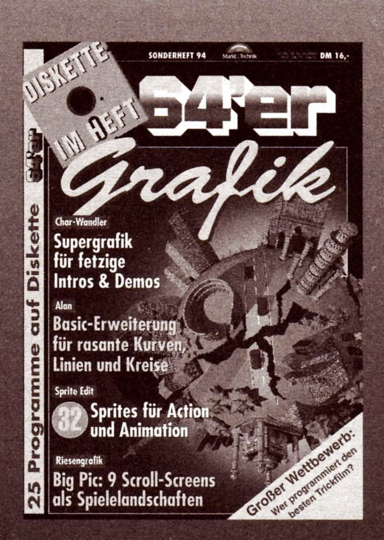
Aus den Erfahrungen der bisherigen Sonderhefte mit Diskette wollen wir ein paar Tips an Sie weitergeben:

1 Bevor Sie mit den Programmen auf der Diskette arbeiten, sollten Sie unbedingt eine Sicherheitskopie der Diskette anlegen.
Verwenden Sie dazu ein beliebiges Kopierprogramm, das eine komplette Diskettenseite dupliziert.

2 Auf der Orginaldiskette ist wegen der umfangreichen Programme nur wenig Speicherplatz frei. Dies führt bei den Anwendungen, die Daten auf die Diskette speichern, zu Speicherplatzproblemen. Kopieren Sie daher das Programm, mit dem Sie arbeiten wollen, mit dem File-Copy-Programm auf eine leere formatierte Diskette und nutzen Sie diese als Arbeitsdiskette.

3 Die Rückseite der Originaldiskette ist schreibgeschützt. Wenn Sie auf dieser Seite speichern wollen, müssen Sie vorher mit einem Diskettenlocher eine Kerbe an der linken oberen Seite der Diskette anbringen, um den Schreibschutz zu entfernen. Probleme lassen sich von vornherein vermeiden, wenn Sie die Hinweise unter Punkt 2 beachten.

ALLE PROGRAMME
aus diesem Heft



HIER

Chefredakteur: Georg Klinge – verantwortlich für den redaktionellen Teil
Stellv. Chefredakteur: Arnd Wängler (aw)
Textchef: Jens Maasberg
Redaktion: Harald Beiler (bl), Jörn-Erik Burkert (lb)
Producer: Andrea Pfliegensdörfer
Redaktionsassistenz: Helga Weber

So erreichen Sie die Redaktion:
Tel. 0 89/46 13-2 02, Telefax: 0 89/46 13-50 01, Btx: 64 064

Manuskripteinsendungen: Manuskripte und Programmlistings werden gerne von der Redaktion angenommen. Sie müssen frei sein von Rechten Dritter. Sollten sie auch an anderer Stelle zur Veröffentlichung oder gewerblichen Nutzung angeboten worden sein, so muß das angegeben werden. Mit der Einsendung von Manuscripten und Listings gibt der Verfasser die Zustimmung zum Abdruck in den von der Markt & Technik Verlag AG herausgegebenen Publikationen und zur Vervielfältigung der Programmlistings auf Datenträgern. Mit Einsendung von Bauanleitungen gibt der Einsender die Zustimmung zum Abdruck in von Markt & Technik Verlag AB verlegten Publikationen und dazu, daß die Markt & Technik Verlag AG Geräte und Bauteile nach der Bauanleitung herstellen läßt und vertreibt oder durch Dritte vertreiben läßt. Honorare nach Vereinbarung. Für unverlangt eingesandte Manuskripte und Listings wird keine Haftung übernommen.

Layout: Isabell Schröfl
DTP-Operator: Heribert Huber
Fotografie: Roland Müller
Titelgestaltung und -grafik: Wolfgang Berns
Computergrafik: Horst Kolodziejczyk

Anzeigenleitung: Peter Kusterer
Anzeigenverwaltung und Disposition: Christoph Mark (421)

Anzeigenauslandsvertretung:
Großbritannien und Irland: Smyth International, Telefon 00 44/8 13 40-50 58, Telefax 00 44/8 13 41-96 02
Niederlande und Belgien: Insight Media, Telefon 00 31/2 15 31 20 42, Telefax 00 31/2 15 31 05 72
Italien: Medias International, Telefon 00 39/31 75 14 94, Telefax 00 39/31 75 14 82
USA und Kanada: M & T International Marketing, Telefon 0 01/41 53 58-95 00, Telefax 0 01/41 53 58-97 39
Japan: Media Sales Japan, Telefon 00 81/3 35 04-19 25, Telefax 0 08 86-2-7 15 19 50
Taiwan: Acer TWP Corporation, Telefon 0 08 86-2-7 13 69 59, Telefax 0 08 86-2-7 15 19 50
Korea: Young Media Inc., Telefon 00 82-2-7 56 48 19, Telefax 00 82-2-7 57 57 89
Israel: Baruch Schaefer, Telefon 0 09 72-3-5 56 22 56, Telefax 0 09 72-3-5 56 69 44
International Business Manager: Stefan Grajer 0 89/46 13-6 38

So erreichen Sie die Anzeigenabteilung:
Tel. 0 89/46 13-9 62, Telefax: 0 89/46 13-7 91

Leiter Vertriebsmarketing: Benno Gaab

Vertrieb Handel: MZV, Moderner Zeitschriften Vertrieb GmbH & Co. KG, Breslauer Straße 5, Postfach 11 23, 85386 Eching, Tel. 0 89/31 90 06-0

Verkaufspreis: Das Einzelheft kostet DM 16,–

Druck: SOV. Graphische Betriebe, Laubanger 23, 96052 Bamberg

Urheberrecht: Alle im 64'er Sonderheft erschienenen Beiträge sind urheberrechtlich geschützt. Alle Rechte, auch Übersetzungen und Zweitverwertung, vorbehalten. Reproduktionen, gleich welcher Art, ob Fotokopie, Mikrofilm oder Erfassung in Datenverarbeitungsanlagen, nur mit schriftlicher Genehmigung des Verlags. Aus der Veröffentlichung kann nicht geschlossen werden, daß die beschriebene Lösung oder verwendete Bezeichnung frei von gewerblichen Schutzrechten sind.

Haftung: Für den Fall, daß im 64'er Sonderheft unzutreffende Informationen oder in veröffentlichten Programmen oder Schaltungen Fehler enthalten sein sollten, kommt eine Haftung nur bei grober Fahrlässigkeit des Verlags oder seiner Mitarbeiter in Betracht.

Sonderdruck-Dienst: Alle in dieser Ausgabe erschienenen Artikel können für Werbezwecke in Form von Sonderdrucken hergestellt werden. Anfragen an Klaus Buck, Tel. 0 89/46 13-1 80, Telefax 0 89/46 13-2 32

© 1993 Markt & Technik Verlag Aktiengesellschaft

Vorstand: Carl-Franz von Quadt (Vorsitzender), Dr. Rainer Doll

Verlagsleiter: Wolfram Höfler
Produktionschef: Michael Koeppe

Direktor Zeitschriften: Michael M. Pauly

Anschrift des Verlags: Markt & Technik Verlag Aktiengesellschaft, Postfach 1304, 85531 Haar bei München, Telefon 0 89/46 13-0, Telex 52 20 52, Telefax 0 89/46 13-1 00

Diese Zeitschrift ist auf chlorfrei gebleichtem Papier mit einem Altpapieranteil von 30 % gedruckt. Die Druckfarben sind schwermetallfrei.

ISSN 0931-8933

Copyright-Erklärung

Name: ....

Anschrift: ....

Datum: ....

Computertyp: ....

Benötigte Erweiterung/Peripherie: ....

Datenträger: Kassette/Diskette ....

Programmart: ....

Ich habe das 18. Lebensjahr bereits vollendet

...., den ....

....
(Unterschrift)

Wir geben die Erklärung für unser minderjähriges Kind als dessen gesetzlicher Vertreter ab.

...., den ....

....

....

Bankverbindung:

Bank/Giroamt: ....

Bankleitzahl: ....

Konto-Nummer: ....

Inhaber des Kontos: ....

Das Programm/ die Bauanleitung: ....

das/die ich der Redaktion der Zeitschrift 64'er übersandt habe, habe ich selbst erarbeitet und nicht, auch nicht teilweise, anderen Veröffentlichungen entnommen. Das Programm/die Bauanleitung ist daher frei von Rechten anderer und liegt zur Zeit keinem anderen Verlag zur Veröffentlichung vor. Ich bin damit einverstanden, daß die Markt & Technik Verlag AG das Programm/die Bauanleitung in ihren Zeitschriften oder ihren herausgegebenen Büchern abdruckt und das Programm/die Bauanleitung vervielfältigt, wie beispielsweise durch Herstellung von Disketten, auf denen das Programm gespeichert ist, oder daß sie Geräte und Bauelemente nach der Bauanleitung herstellen läßt und vertreibt bzw. durch Dritte vertreiben läßt. Ich bin auch mit einer Zweitverwertung, z.B. in Form von Sonderdrucken einverstanden. Ich erhalte, wenn die Markt & Technik Verlag AG das Programm/die Bauanleitung druckt oder sonst verwertet, ein Pauschalhonorar.

Disklader – Programme laden mit Komfort

Diskettenoberfläche de Luxe

Keine umständlichen Ladeanweisungen und ein übersichtliches Inhaltsverzeichnis der Diskette auf dem Bildschirm. Unser »Disklader« erfüllt auch gehobene Ansprüche.

Entwicklungshelfer sind gefragt, denn noch immer sind einige Arbeitsschritte nötig, um beim C64 ein Inhaltsverzeichnis von der Diskette zu erhalten. Außerdem erschweren manche Unterdateien zu einem Programm die Übersicht im »Directory«. Genau hierfür finden Sie einen »Feuerwehrmann« auf der ersten Seite der beiliegenden Diskette – den »Disklader«. Er generiert eine Benutzeroberfläche für Ihren C 64. Darin sind Funktionen integriert, wie:

- Anwahl einzelner Programme (mit jeweiliger Kurzbeschreibung),
- automatisches Laden und Starten von Diskette oder
- Erkennung der richtigen Diskette bzw. Diskettenseite.

Da sich der Disklader an erster Stelle auf der Diskette zum Sonderheft befindet, genügt es, zum Laden einzugeben:

```
LOAD":*",8
```

Nach der Bestätigung mit <RETURN> dauert es ca. 15 s, bis die Datei im Speicher ist. Sie starten mit RUN und <RETURN>. Anschließend wird das File entpackt (ca. 2 s) und es erscheint die Benutzeroberfläche des »Disklader« (s. Abbildung). In der rechten unteren Bildschirmhälfte sehen Sie weiß umrandet den Namen des ausgewählten Programms. Die unterste Bildschirmzeile ist die dazugehörige Kurzerklärung. Zusätzlich finden Sie in der rechten unteren Bildschirmhälfte den Text »Seite 1 auf Disk« oder »Seite 2 auf Disk«. Da Sie die Inhaltsverzeichnisse beider Seiten (ohne die Disk zu wenden) durchblättern können, finden Sie hier den Hinweis, auf welcher Diskettenseite sich das gewählte Programm befindet. Durch Tastendruck <CRSR aufwärts> bzw. <CRSR abwärts> wählen Sie das nächste oder vorherige Programm. Sie blättern quasi durch den Inhalt der Programme. <HOME> bringt Sie zum ersten Eintrag des Inhaltsverzeichnisses. Selbstverständlich sind nur die Programme verzeichnet, die sich eigenständig laden oder starten lassen.



Kurzinfo: Disklader

Programmart: Hilfsprogramm zum Laden der Programme auf der beiliegenden Diskette
Laden: LOAD":*",8
Starten: nach dem Laden mit RUN
Steuerung: Tastatur
Programmautor: H. Großer

<RETURN> führt Sie in den Ladeteil. Ist kein Diskettenfehler aufgetreten, erscheint kurzzeitig »00,OK,00,00« am Bildschirm. Eventuelle Fehleranzeigen bleiben sichtbar am Bildschirm (z.B. »21,READ ERROR,18,00«= Drive not ready). Sie lassen sich durch einen beliebigen Tastendruck wieder löschen. Schlagen Sie bitte vorher im Handbuch Ihrer Floppy nach und beseitigen Sie den Fehler. Eine andere Art der Fehlermeldung wird durch einen blinkenden Text dargestellt (z.B.»Bitte Disk wenden« »Falsche Diskette«). Sind Fehler ausgeblieben, lädt der Disklader das von Ihnen gewählte Programm von der Diskette und startet es. Ladefehler, die in dieser Phase auftreten, werden nicht mehr berücksichtigt: Der Disklader wird vom neuen Programm einfach überschrieben. Sonst könnten wir nur Programme veröffentlichen, die mit der Benutzeroberfläche zusammenarbeiten. Bei vielen Spielen, Tricks oder Tools ist dies aber nicht der Fall.

Für Sie bedeutet dies, nach jedem Starten eines Programms den »Disklader« erneut zu laden. Wer die Benutzeroberfläche verlassen will, gibt <RUN/STOP> ein. Sie befinden sich dann im normalen »Basic« des C 64. Für einen Neustart befehlen Sie

```
SYS 12032
```

und bestätigen mit <RETURN>. Dieser Neustart funktioniert auch nach einem Reset, d.h. wenn Sie durch den entsprechenden Taster einen Hardware-Reset ausgelöst haben. Allerdings dürfen Sie in der Zwischenzeit kein neues Programm laden, da dies den verwendeten Speicherbereich meist überschreibt. Laden Sie in diesem Fall den Disklader neu.

Wir haben bei der Programmierung größten Wert auf Kompatibilität mit den unterschiedlichsten Systemerweiterungen gelegt. Lediglich bei der Gerätekonfiguration C 128 mit RAM-Erweiterung und zweiter Diskettenstation sollten Sie die externe Floppy ausschalten. (gr)

Grafik – leicht verständlich

Reise ins Pixel-Land

Sie faszinieren jeden, die grafischen Möglichkeiten des C 64. Mit ein paar POKEs läßt sich jeder Textbildschirm in eine hochauflösende Grafiklandkarte (Bitmap) verwandeln – auch in Basic 2.0.

Was bedeutet hohe Auflösung? Der normale Textbildschirm bietet nach dem Einschalten des Computers 1000 markante Speicherstellen, die man entweder ein- oder ausschalten kann: 25 Zeilen x 40 Spalten = 1000 Plätze. In jedem Bildpunkt steht ein Zeichen (Charakter) des aktiven Zeichensatzes. Das können Buchstaben, Zahlen, Grafikzeichen von der Tastatur – oder Leerzeichen sein (dann ist der Bildschirm gelöscht!). Beachten Sie die Tabelle der Bildschirmcodes im Handbuch des C 64.

Eines haben alle Zeichen gemeinsam: Sie bestehen aus einer Matrix von acht Bytes mit jeweils acht Bits, also 64 Bildpunkten (Pixel). Der Computer verfügt in einer Tabelle im Betriebssystem über 255 verschiedene Codes pro Zeichenmatrix. Je nachdem, welcher Wert angegeben ist, erscheint das betreffende 8 x 8-Pixel-Muster an gewünschter Cursor-Position auf dem Bildschirm. Direkt beeinflussen lassen sich diese Pixel nur beim Ändern des Zeichensatzes – und da auch nur Byte für Byte pro Zeichen.

In einer speziellen Speicherstelle gibt's einen Schalter, der den Anzeigemodus des Computers verändert: Bit 5 in $D011 (53265). Ist es aktiviert (Wert 32), benutzt der Computer die »High Resolution« (Hires). Aus 1000 Bildpunkten sind nun 64mal soviel geworden:

8 Bit x 8 Byte x 1000 Bildstellen = 64 000

Jetzt können Sie statt einer 8 x 8-Punkte-Matrix jedes einzelne Bit in diesem Feld beeinflussen: 1 = Bit eingeschaltet, 0 = gelöscht. Als Berechnungsgrundlage dient die vom Textmodus bekannte Auflösung: 40 Spalten zu jeweils acht Bit ergeben 320 Pixel in horizontaler (x-) Richtung, 25 Zeilen pro acht Byte registrieren 200 vertikale Bildpunkte (y-Richtung). Daran erkennt man, daß hochauflösende Grafik sehr speicheraufwendig ist: 64 000 Bits benötigen 8000 Adressen im Computer (64 000 : 8).

»Bitmapping« ist die meist verwendete Grafiktechnik beim Computer (nicht nur beim C 64!), um detaillierte Bilder wiederzugegeben (vergleichbar mit dem gerasterten Bild in einer Zeitung). Neben der Standard-Bitmap des C 64 gibt's noch den **Multicolormodus**, der zwar vier verschiedene Farben bietet, dies aber mit einer geringeren Auflösung bezahlen muß: nur noch 160 Pixel in horizontaler Richtung. Wir zeigen Ihnen Schritt für Schritt, wie man diese Bitmaps in Basic erzeugt und aktiviert.

Dies ist zwar prinzipiell eine Aufgabe für Maschinensprache, denn Grafikbefehle gibt's im Basic 2.0 des C 64 überhaupt nicht (dazu braucht man schon so mächtige Basic-Erweiterungen wie auf der Diskette zu diesem Sonderheft) ... Trotzdem: Es geht, mit den entsprechenden Rechenroutinen und unter Einsatz vieler Variablen!

Schalten wir zunächst die Hires-Bitmap ein:

```
10 POKE 53265, PEEK(53265) OR 32
```

Bit 5 wurde durch die OR-Verknüpfung aktiviert. Nach dem Start dieser Basic-Zeile mit RUN erscheint jedoch nur wirres Byte-Chaos auf dem Bildschirm. Drücken Sie nun die Tastenkombination <RUN/STOP RESTORE> oder geben Sie »blind« ein:

```
POKE 53265, PEEK(53265) AND 223
```

Damit ist der Originalzustand wiederhergestellt. Das Gewirr, das Sie gesehen haben, waren die ersten 8000 Bytes des C 64 (Speicherstellen 0 bis 7999), denn exakt dort liegt nun die Bitmap. Das bringt Probleme: Ändern wir dort irgendwelche Bits oder Bytes, dann verändern wir eventuell wichtige Speicheradressen der Zeropage (0 bis 255), des Speicherstapels (Stack, von 256 bis 511), der Betriebssystemvektoren ab 768 – und last not least unseres Basic-Programms, das ab Adresse 2049 beginnt. Die Lage unserer Grafiklandkarte ist also alles andere als günstig ... aber wo soll man sie plazieren und wie läßt sich das umstellen? Die Frage nach dem »Wo« ist schnell beantwortet: innerhalb eines Bereichs von 16 KByte (= 16384 Byte), den der Videochip VIC-II überblicken kann. Da die ersten 8000 Byte aus den genannten Gründen nicht in Frage kommen, sind's eben die zweiten: ab Adresse 8192 bis 16191. Teilen wir die Anfangsadresse durch »1024« und POKEn das Ergebnis (8) in die Speicherstelle 53272:

```
20 POKE 53272, PEEK(53272) OR 8
```

Da der Beginn der Bitmap (8192) bei unseren weiteren Berechnungen noch eine Rolle spielen wird, speichern wir diese Zahl am besten in einer Variablen:

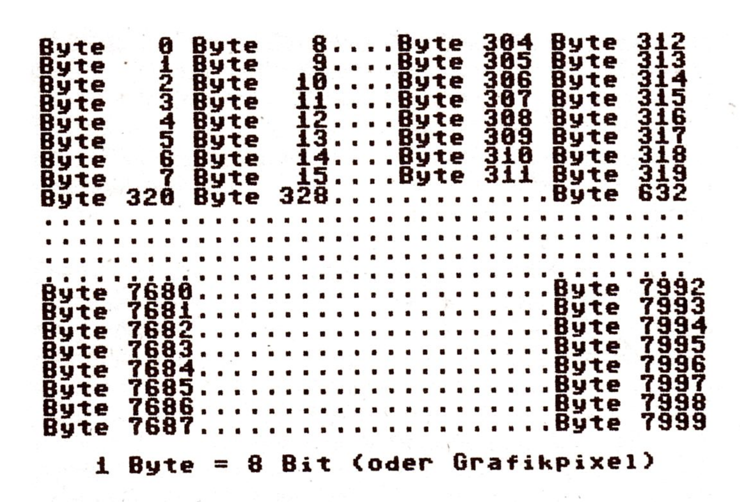
```
5 BA = 8192
20 POKE 53272, PEEK(53272) OR BA/1024
```

Um das Byte-Gewirr auf dem Bildschirm (dort sieht man die aktuellen, zufälligen Inhalte der Adressen 8192 bis 16191) abzustellen, muß man diesen Bereich löschen (Null-Bytes eintragen!):

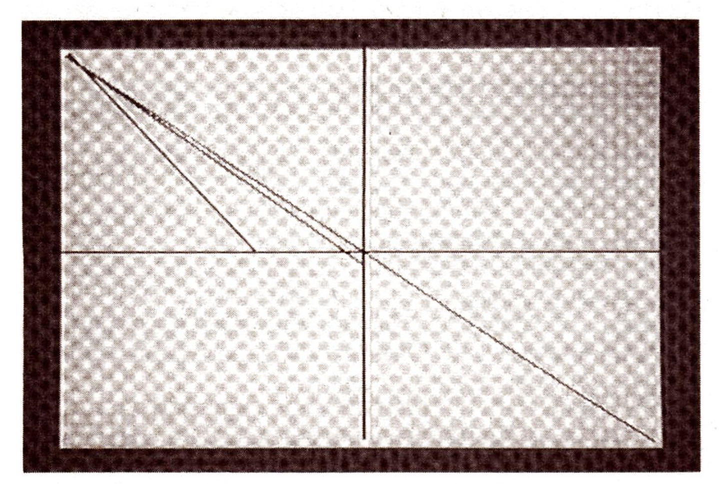
```
30 FOR I=BA TO BA+7999
40 POKE I,0
50 NEXT I
```

Nach RUN kann man beobachten, wie sich der Screen vom Chaos befreit. Ganz stimmt das nicht: Es bleiben noch immer einige undefinierbare Farbblöcke zurück – sie haben aber mit der Hires-Bitmap nichts zu tun, sondern mit dem Farbspeicher für die Grafiklandkarte. Diese Farben kommen nun nicht mehr aus den Adressen 55296 bis 56295 (wie im Textmodus), sondern wurden vom Computer automatisch nach 1024 bis 2023 verlegt (im Textmodus ist das der Bildschirmspeicher). Da es aber nach wie vor nur 1000 Byte bleiben, muß man sich damit abfinden, daß sich trotz hochauflösender Grafik immer nur 8 x 8 Pixel verschiedenfarbig anzeigen lassen: Pro Pixel eine eigene Farbe, das würde nochmals 64 000 Byte erfordern (die der C 64 nicht hat).

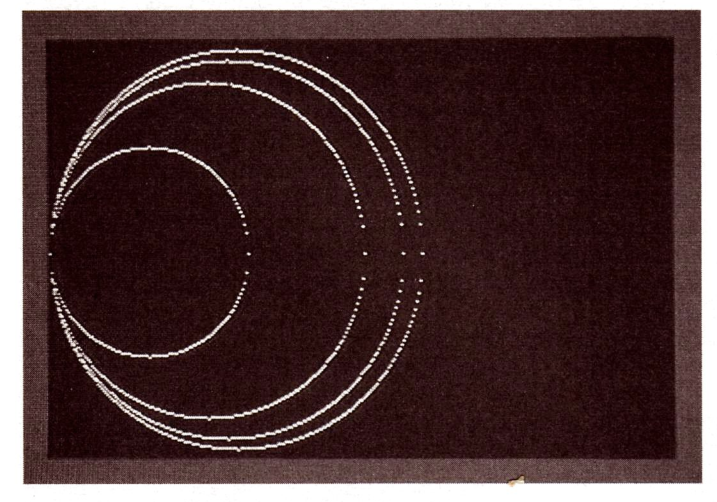
Wie stellt man Farbe ein? Bei der Standard-Bitmap gibt's nur zwei Farben: Zeichenfarbe (Vordergrund) und Hintergrund. Die Rahmenfarbe wird nach wie vor durch die Speicherstelle 53280 bestimmt (wie schon aus dem Textmodus bekannt). Das Farb-Byte muß man nun in zwei Hälften teilen: Die oberen vier Bit (deren Gesamtwert multipliziert man mit »16«) sind für die Vordergrundfarbe, die unteren vier Bit (sie behalten ihren realen Wert) für den Hintergrund zuständig. Ein Beispiel: Sie möchten eine gelbe Hintergrundfarbe mit blauen Gra-



[1] Die Aufteilung der Grafik-Bytes im Hires-Modus



[2] Linien in beliebiger Länge per Basic-Routine



[3] Kreise erzeugt man per SQR-Anweisungen

fikpixeln. Gelb hat den Farbcode »7«, Blau die Nummer »6«. Die Rechenformel sieht dann so aus (C ist z.B. die Farbvariable):

```
55 C = 16 * 6 + 7
```

Um diese Farbwahl allen 1000 Speicherplätzen des Hires-Farb-RAM mitzuteilen, müssen wir wieder eine Schleife benutzen:

```
60 FOR I=1024 TO 2023
70 POKE I,C
80 NEXT I
```

Schneller als beim Löschen des Hires-Bereichs füllt sich nun der Bildschirm gelb: es sind nur 1000 Speicherstellen. Um ein Pixel zu setzen oder zu löschen, muß der Computer natürlich wissen, wie er das richtige Bit im Speicher findet. Das bedeutet: Er muß das zu ändernde Zeichen, die Zeichenreihe und das entsprechende Bit in dieser Reihe identifizieren. Für den C 64 existiert nämlich theoretisch noch immer der Textbildschirm, der mit 25 Zeilen und 40 Spalten vollgeschrieben ist: nur läßt er im Hires-Modus zu, daß der Speicherbereich anders interpretiert wird (Abb.1).

Um die exakte Position eines Bildpunkts zu fixieren, gibt's bestimmte Rechenfunktionen, die man numerischen Variablen zuweist. Den Beginn des Hires-Bildschirms haben wir bereits mit BA = 8192 festgelegt. Die Position in der Bildschirmzeile errechnet sich so:

```
RO = INT(Y/8)
```

Das dazugehörige Zeichen (Charakter) wird mit dieser Formel berechnet:

```
CH = INT(X/8)
```

Die Byte-Zeile innerhalb dieser Zeichenmatrix:

```
LI = Y AND 7
```

Das entsprechende Bit aus diesem Byte:

```
BI = 7-(X AND 7)
```

Alle Formeln setzt man nun zusammen. Die Byte-Zeile, in der das Pixel liegt, kann man jetzt eingrenzen:

```
BY = BA + RO * 320 + CH * 8 + LI
```

Endlich ist es soweit! Der gewünschte Bildpunkt erscheint mit dieser POKE-Anweisung auf der Grafik-Bitmap:

```
POKE BY, PEEK(BY) OR 2BI
```

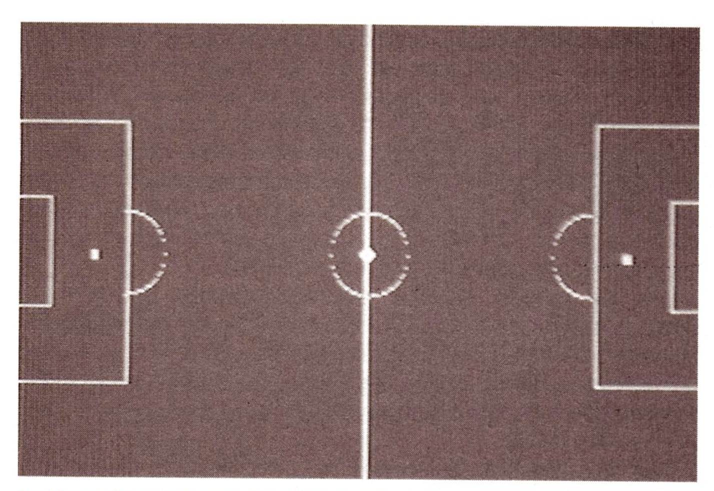
Dazu gibt's fünf Demoprogramme:

Setpixel zeigt, wie man Bitmaps einschaltet, das Farb-RAM intialisiert und nach Eingabe der x- und y-Koordinaten einen Bildpunkt in der Hires-Grafik erzeugt. Da der Grafikbildschirm nur das erste Mal gelöscht wird, bleibt das eingeschaltete Pixel erhalten. Wenn Sie per Tastendruck zurück zur nächsten Koordinateneingabe schalten, lassen sich so beliebige Grafiken entwerfen. Allerdings ist es ziemlich mühsam, da man Punkt für Punkt eintragen muß.

Draw geht bereits einen Schritt weiter: Damit kann man ganze Linien auf der Bitmap fabrizieren (Abb.2). Sie müssen lediglich Start- und Endkoordinaten der Linie angeben, Länge und Bildpunkte dazwischen berechnet das Programm automatisch und setzt die Pixel an entsprechender Stelle. Es wird eine Schleife gebildet, deren Laufvariablenwert sich aus der Differenz zwischen Start und Ende berechnet. Da bei diesem Beispielprogramm vor jedem neuen Durchlauf die Bitmap nicht gelöscht wird, kann man problemlos mehrere Linien eintragen und z.B. geometrische Körper erzeugen.

Box motzt unser Beispielprogramm noch ein bißchen weiter auf: umständliche Koordinateneingaben entfallen. Sie sind bereits als Datenzeilen im Programm abgelegt und werden durch den READ-Befehl wieder in den Variablenspeicher geholt. Man erkennt auf dem Hires-Screen Rechtecke unterschiedlicher Größe. Die Zeichengeschwindigkeit läßt allerdings – bedingt durch Basic – zu wünschen übrig!

Was wäre Hires-Bitmap ohne Kreise und Ellipsen? **Circle** zeigt, daß sich auch solche Grafikkörper mit Basic-Befehlen realisieren lassen. Wer glaubte, mit dem Radiuswert und der Ludolfschen Zahl Pi (die Konstante 3,14159265, die bei Kreisberechnungen eingesetzt wird) arbeiten zu müssen, sieht sich eines Besseren belehrt: Der Programmierer benutzt zur Berechnung die SQR-Funktion des Basic 2.0 (Quadratwurzel aus beliebiger Zahl). Durch geschickte Programmierung wird so, je nach Koordinatenwert, ein Halbkreis auf dem Bildschirm erzeugt. Durch Vertauschen der Vorzeichen in der Rechenroutine zeichnet der Computer je einen Halbkreis auf dem unteren und oberen Bildschirm (zusammengefügt ergibt's den ganzen, Abb. 3).



[4] Fußballstadion aus der Vogelperspektive

Alle Grafikelemente (Linie, Rechteck und Kreis) vereint das letzte Programmbeispiel zur Hires-Grafik in Basic 2.0: **Stadion**, ein Fußballfeld von oben (Abb. 4). Jetzt fehlen nur noch die Sprites der Fußballer, und schon kann die Bundesliga-Saison starten ... (bl)

Vier Farben: geringere Auflösung

Zur Standard-Bitmap im hochauflösenden Grafikmodus gibt's eine Alternative: die Multicolor-Bitmap. Damit lassen sich in einem 8 x 8-Pixelbereich vier verschiedene Farben aktivieren: Hintergrundfarbe, Vordergrundfarbe (wie bei der Standard-Bitmap), Multicolorfarbe 1 und 2. Dadurch verringert sich aber die Auflösung: horizontal kann man jetzt nur noch maximal 160 Pixel eintragen! Jeder Bildpunkt erscheint nun als ein Doppelpixel, da das zweite Bit die Art der Farbinformation speichert. Die Farben werden jetzt durch Speicherstellen des VIC-II-Chip bestimmt, lediglich die Bitkombination der Adressen im Farb-RAM (1024 bis 2023) gibt Auskunft darüber, woher die aktuelle Farbe kommt:

- Doppelbit: binär 00 (Hintergrundfarbe aus Adresse 53281),
- Doppelbit: binär 01 (Multicolorfarbe 1, Bits # 4 bis 7 des Hires-Farb-RAM ab 1024),
- Doppelbit: binär 10 (Multicolorfarbe 2, Bits # 0 bis 3 des Hires-Farb-RAM),
- Doppelbit: binär 11 (Vordergrundfarbe gemäß Farb-RAM für den Textmodus ab 55296).

Das Mehrfarben-Bitmapping wird durch die Speicherstelle 53270 bestimmt (Bit 4). Wenn Sie unsere Grafikbeispiele in Basic durch die Multicolorbrille betrachten möchten, müssen Sie jedes Programm um folgende Basic-Zeilen ergänzen:

```
1015 POKE 53270, PEEK(53270) OR 16
2015 POKE 53270, PEEK(53270) AND 239
```

Soll sich die Hintergrundfarbe ebenfalls ändern, müssen Sie den gewünschten Code in Adresse 53281 POKEn. (bl)

Grafikbanken

Unsere Basic-Tips zur Aktivierung hochauflösender Grafik gelten nur unter der Voraussetzung, daß Ihr Basic-Programm nicht länger als ca. 6000 Bytes ist: Sonst kommt es mit der Hires-Bitmap in Konflikt.

Eng begrenzt ist der Speicherplatz von VIC-Bank 1: Zeropage, Basic-Programm, Variablen und Bitmap, zusammengepfercht auf 16 KByte. Auf der Suche nach einem ungestörten Plätzchen stößt man auf die idealere VIC-Bank 2: von 16384 bis 32767. Man verlegt die Grafiklandkarte in den zweiten Teil der Bank (ab 24576) und verschiebt den Bildschirm ebenfalls (1024 Speicherstellen darunter, also ab 23552). Diesen Plan muß man dem Computer mitteilen:

```
POKE 56576, (PEEK(56576) AND 252) OR 2
POKE 56578, PEEK(56578) OR 3
```

Diese beiden Adressen im Speicherbaustein CIA II sind dafür zuständig, Ihren Plan in die Tat umzusetzen. Zwei weitere POKEs sind notwendig:

```
POKE 53272, 120: POKE 648,92
```

Wenn man Bit 3 der Adresse 53272 einschaltet (= 1), weiß der VIC-Chip, daß sich die Bitmap in der zweiten Hälfte befindet. Die Bits 4 bis 7 teilen dem Computer mit, wohin man den Bildschirmspeicher ausgelagert hat. Außerdem muß man das Betriebssystem über Speicherstelle 648 informieren, ab welchem Speicherplatz die Bildschirmdaten liegen (23552 : 256 = 92). Zwar ist die Bitmap jetzt ziemlich weit oben im Speicher (Vorteil: Basic-Programme können nun erheblich länger sein, mehr als 20 000 Bytes stehen zur Verfügung), aber dennoch im Basic-RAM. Die Grafik könnte durch Variablen überschrieben werden. Das stellen wir schleunigst ab: Gaukeln Sie dem C 64 vor, sein Speicher sei kleiner als gewohnt:

```
POKE 51,0: POKE 52,92
POKE 55,0: POKE 56,92
```

Diese Zeropage-Adressen sind die Zeiger auf das Ende der Variablenfelder und des gesamten Variablenspeichers: Jetzt erstreckt er sich nur noch bis 23552 (92 X 256).

Bekannt ist inzwischen auch, wie man Grafikpixel berechnet und auf dem Bildschirm plaziert. So sehr häufiger Variableneinsatz in Basic-Programmen zu empfehlen ist: Sie bremsen den Ablauf der Rechen- und Pixelsetzroutine erheblich. Einfacher und schneller geht's, den gesamten Vorgang in vier Basic-Zeilen zu definieren:

```
BA = 24576
BY = (X AND 504)+40*(Y AND 248)+(Y AND 7)
BI = 7-(X AND 7)
POKE BA+BY, PEEK(BA+BY) OR (2BI)
```

Wenn Sie nun die neue Routine mit der im Listing »Draw« vergleichen: das Ziehen einer durchgehenden Linie auf dem Grafikbildschirm dauert jetzt sechs Sekunden weniger!

Unendlich langsam dagegen ist das Löschen der Bitmap per Basic-Schleife. Hier hilft wirklich nur Maschinensprache. Die entsprechende Routine finden Sie im Grafikbeispiel **Speedy Hires** auf unserer Sonderheftdiskette, das die neue Routine zum Setzen von Bildpunkten verwendet. Die Zeilen 1000 bis 1030 enthalten die Maschinensprache-Daten und die Einleseschleife. Das Utility liegt im Speicher ab Adresse 49152 und läßt sich in eigenen Programmen verwenden. (bl)

Neuer Zeichensatz gefällig?

So einfach, wie sich's anhört, ist es leider nicht: Der C 64 holt seine Bildschirmzeichen (Characters) aus dem Zeichensatz-ROM ab $D000 (53248 bis 57343). ROM (Read Only Memory) ist ein Speicherbereich, den man nur lesen – aber nicht per POKE beschreiben kann. Wer seinen Computer z.B. mit deutschen Umlauten, Sonderzeichen oder Mauersteinen für einen Spiele-Level ausstatten will, muß den Zeichensatzbereich an eine passende Stelle ins RAM kopieren (in VIC-Bank 1 liegt der ideale Bereich ab Adresse $3000, 12288 dez.) und die Zeiger in den entsprechenden Speicherstellen darauf richten. Diese Basic-Schleife macht's:

```
FOR I=0 TO 4095: POKE 12288+I,
PEEK(53248+I): NEXT
POKE 53272, PEEK(53272) OR 12
```

Gehen Sie ruhig inzwischen Kaffee trinken, denn das dauert ... mehr als eine halbe Minute. Um so größer ist die Ent-

täuschung, wenn Sie wiederkommen: Lediglich die Bytes des VIC-, des SID- und der CIA-Chips wurden ins RAM übertragen (ganz zu schweigen vom Farb-RAM ab Adresse 55296), also der Ein-Ausgabe-Bereich (I/0-Register). Dieser Platz ist im C 64 gleich dreimal belegt:
- I/0-Bereich,
- Zeichensatz und
- freies RAM.

Welche Konfiguration aktiv ist, bestimmt Adresse 1. Der Normalinhalt dieser Speicherstelle nach dem Einschalten des Computers ist »55« oder %0011011 (binär). Unsere Tabelle 1 macht die Funktionen der Bitbelegungen deutlich (nur die ersten drei sind relevant, »1« = Bit ein-, »0« = Bit ausgeschaltet).

Es muß also Bit 2 (Wertigkeit: 4) gelöscht werden, um das Zeichensatz-ROM zu erwischen: POKE 1,51. Und schon tritt das nächste Problem auf: Durch diesen POKE wurde der I/O-Bereich abgeschaltet – der Computer gehorcht keiner Eingabe mehr! Schuld daran ist der Interrupt, der den C 64 alle 1/60stel Sekunden für spezielle Aufgaben unterbricht (Tastatur abfragen, Bildschirm erneuern usw.). Wenn Sie diese Unterbrechungen auch noch abschalten (Speicherstelle 56334 auf »0« setzen), dann klappt's:

Speicherstelle 1 (Bitbelegung)

Bit.Nr.	Zustand	Funktion
0	1	Basic-ROM (40960 bis 49151)aktiv
	0	Basic-RAM eingeschaltet
1	1	Kernel-ROM (57344 bis 65535) ein
	0	RAM unter Kernel ein
2	1	I/O-Register (53248 bis 57343) ein
	0	Zeichensatz-ROM ein

Die Bits 3 bis 5 kümmern sich um den Datasettenbetrieb, Nr. 6 und 7 werden nicht benutzt (also immer »0«). **Tabelle 1**

```
10 POKE 56334,0: POKE 1,51
20 FOR I=0 TO 4095
30 POKE 12288+I, PEEK(53248+I)
40 NEXTI
50 POKE 1,55: POKE 56334,1
60 POKE 53272, PEEK(53272) OR 12
```

Zeile 60 bedarf einer Erläuterung: die unteren vier Bit der Adresse 53272 enthalten den Zeigerwert für den Beginn des Zeichensatzbereichs, den der Computer verwenden soll (man muß die Zahl mit »1024« multiplizieren, dann kommt »12288« raus). Jetzt können Sie die Bytes im RAM nach Belieben ändern und ummodeln.

Auch dieses Verfahren ist kaum ideal: Abgesehen von der Übertragungsdauer (30 Sekunden) bleibt Ihnen auch hier nicht viel Platz für Ihr Basic-Programm: knappe 12 000 Bytes. Wie im Beispielprogramm Speedy Hires empfehlen wir, die VIC-Bank 2 einzuschalten und z.B. dort einen fertig geänderten Zeichensatz an die relative Adresse 12288 (erhöht um »16384«, also »28672«) zu laden (POKE-Wert für 53272: 188). Der Bildschirmspeicher muß ebenfalls relativ verschoben werden (am besten 1000 Bytes darunter): 27648. Diesen Wert, geteilt durch »256« müssen Sie zusätzlich in die Adresse 648 POKEn: 108 (Normalinhalt: 4). Damit stehen Ihnen immerhin 25 600 freie Basic-Bytes zur Verfügung. (bl)

Bildschirm- und Zeichensatz-RAM

Jeder, der Demos, Intros oder Spiele programmiert, muß neue Zeichensätze (Charsets) entwerfen und irgendwo im freien RAM unterbringen. Schon geht's los mit der Rechnerei: Das High-Byte der VIC-Adresse $D018 (53272) muß die Kennzahl des neuen Charakter-RAM enthalten; das Low-Byte ist für die Lage des Bildschirm-RAM verantwortlich (das ja in derselben 16-KByte-VIC-Bank liegen muß wie die neuen Zeichen, sonst herrscht Byte-Chaos auf dem Screen!).

Beispiel: Der geänderte Zeichensatz liegt bei $2000 (8192), das Bildschirm-RAM soll nach $1800 (6144) verlegt werden. Die Berechnung sieht so aus:

```
6144/1024 + 8192/1024 * 16
```

Das Ergebnis (134) POKEt man in Adresse 53272.

Einfacher geht's aber, wenn Sie sich an Tabelle 2 halten: die oberste Zahlenreihe enthält die Werte fürs Bildschirm-RAM, die unterste die des entsprechenden Zeichensatzspeichers. Zusätzlich lassen sich die jeweiligen VIC-Bänke ablesen (0 bis 3), in denen die einzelnen RAM-Abschnitte untergebracht sind. Die ermittelte Zahl schreibt man in Adresse $DD00 (56576) – schon ist die richtige VIC-Bank eingestellt! (Sascha Lempke/bl)

Animierte Grafik: Sprites

Kleine Kobolde huschen über den Bildschirm – das sind die Sprites des C 64: 64 Byte große Grafikobjekte, die unabhängig vom aktivierten Bildschirmmodus (Text oder Hires-Grafik) ihr Eigenleben führen und eine beträchtliche Menge der Register des Video-Interface-Chips (VIC) belegen.

Bis zu acht Sprites kann man beim normal konfigurierten

Bildschirmspeicher

$00 (0)	$01 (1)	$02 (2)	$03 (3)	$04 (4)	$05 (5)	$06 (6)	$07 (7)	$08 (8)	$09 (9)	$0A (10)	$0B (11)	$0C (12)	$0D (13)	$0E (14)	$0F (15)
							Grafik Speicher 1 $00(0)								
0 $0000	1024 $0400	2048 $0800	3072 $0C00	4096 $1000	5120 $1400	6144 $1800	7168 $1C00	8192 $2000	9216 $2400	10240 $2800	11264 $2C00	12288 $3000	13312 $3400	14336 $3800	15360 $3C00
							Grafik Speicher 2 $01(1)								
16384 $4000	17408 $4400	18432 $4800	19456 $4C00	20480 $5000	21504 $5400	22528 $5800	23552 $5C00	24576 $6000	25600 $6400	26624 $6800	27648 $6C00	28672 $7000	29696 $7400	30720 $7800	31744 $7C00
							Grafik Speicher 3 $02(2)								
32768 $8000	33788 $8400	34812 $8800	35836 $8C00	36860 $9000	37884 $9400	38908 $9800	39932 $9C00	40956 $A000	41980 $A400	43004 $A800	44028 $AC00	45052 $B000	46076 $B400	47100 $B800	48124 $BC00
							Grafik Speicher 4 $03(3)								
49152 $C000	50176 $C400	51200 $C800	52224 $CC00	53248 $D000	54272 $D400	55296 $D800	56320 $DC00	57344 $E000	58368 $E400	59392 $E800	60416 $EC00	61440 $F000	62464 $F400	63488 $F800	64512 $FC00
(0) $00	(16) $10	(32) $20	(48) $30	(64) $40	(80) $50	(96) $60	(112) $70	(128) $80	(144) $90	(160) $A0	(176) $B0	(192) $C0	(208) $D0	(224) $E0	(240) $F0

Charset Speicher

Tabelle 2

Bitbelegung der Sprite-Register

Sprite-Nr.	Bit-Nr.	Wertigkeit
0	0	1
1	1	2
2	2	4
3	3	8
4	4	16
5	5	32
6	6	64
7	7	128

Tabelle 3

C 64 gleichzeitig auf dem Screen herumhopsen lassen (mit Tricks, die den Rasterstrahl des Monitors beeinflussen, sind's mehr, z.B. 16, 32 oder 64). Jedes Sprite bekommt eine Nummer von 0 bis 7. Für jede Sprite-Funktion gibt's das entsprechende Register im VIC-Chip (z.B. Nr. 29, um die Sprite-Höhe zu verdoppeln, also in y-Richtung dehnen).

Um die Adressen anzusprechen, muß man die Registernummer zur Basisadresse des VIC addieren, z.B.:

53248 (Basisadresse) + 29 = 53277

Um ein bestimmtes Sprite gezielt anzusprechen, ist bei nahezu allen VIC-Registern eines der acht Bit des jeweiligen Bytes zuständig (s. Tabelle 3). Will man also Sprite Nr. 5 doppelt so hoch erscheinen lassen, muß auch Bit #5 von Adresse 53277 eingeschaltet werden, ohne aber die anderen Sprites zu tangieren (die sollen ihre Normalgröße behalten!). Zum Setzen oder Löschen von Bits innerhalb eines Bytes bedient man sich der Boole'schen Verknüpfungen OR (einschalten) bzw. AND (ausschalten).

Unser Schema zeigt, wie man Bits gezielt manipuliert (NR = Bit- bzw. Sprite-Nummer, RG = VIC-Register):

Bit setzen: POKE RG,PEEK(RG) OR 2NR

Bit löschen: POKE RG,PEEK(RG) AND 255 - (2NR)

Zur Verdoppelung der Sprite-Höhe gilt dann diese Anweisung:

```
POKE VIC+29, PEEK(VIC+29) OR 2NR
```

NR ist die Sprite-Nummer (0 bis 7), VIC die Basisadresse des Grafik-Chips (53248). Wenn Sie's noch zusätzlich verbreitern (in x-Richtung dehnen) möchten, müssen Sie Register 23 manipulieren:

```
POKE VIC+23, PEEK(VIC+23) OR 2NR
```

Ergebnis: ein vierfach vergrößertes Sprite.

Soll's wieder Normalgröße erhalten, kommt die AND-Verknüpfung zum Einsatz:

```
POKE VIC+29, PEEK(VIC+29) AND 255 - 2NR
POKE VIC+23, PEEK(VIC+23) AND 255 - 2NR
```

Weitere Infos zu wichtigen Sprite-Registern finden Sie in unserer Tabelle 4. Sprite-Muster definiert man am besten mit einem Datenblatt (Abb. 5) oder mit unserem »Mini-Sprite-Editor« (s. Beschreibung). Noch bequemer geht's allerdings per professionellem Sprite-Tool (z.B. »Spritemon 2.2« im 64'er-Sonderheft 75 oder »Single-Sprite-Editor« in diesem Heft). (bl)

Mini-Sprite-Editor

Kürzer geht's kaum noch – der Effekt ist dennoch derselbe wie bei umfangreichen Tools: Die entsprechenden Byte-Werte fürs Sprite-Muster erscheinen fix und fertig berechnet rechts neben dem Editorfeld:

```
10 PRINTCHR$(19);
20 FOR I=0 TO 20: A=0: FOR N=0 TO 7:
30 A=A-2(7-N)*(PEEK(1024+40*X+N+8*I)=42)
40 NEXT: A(I+1)=A:NEXT
50 PRINTTAB(25)A(1)A(2)A(3):X=X+1
60 IF X<21 THEN 20
```

Löschen Sie den Bildschirm und zeichnen Sie darauf Ihr Sprite-Muster (Pixel setzen = <*>, löschen = <SPACE>). Fangen Sie in der linken oberen Bildschirmecke an! Bewegen Sie dann den Cursor in die drittletzte Zeile (Nr. 22) und starten Sie den Mini-Editor mit RUN: Im Nu erscheinen die Daten neben dem Muster – man muß sie nur noch notieren (Abb. 6)! (Dr. L. Meyding/bl)

Sprite-Spürhund

Umfangreiche und komfortable Utilities gibt's, um den Speicher des C 64 nach versteckten Sprites abzusuchen. Meist sind solche Programme recht umfangreich und verbrauchen viel Platz auf Disk. Es geht aber auch kürzer:

```
10 V=53248: POKE V,160: POKE V+1,130
20 POKE V+23,255: POKE V+29,255: POKE
V+21,255
30 PRINT CHR$(147);A*64
40 GET A$
50 IF A$=CHR$(29) THEN A=A+1: IF A=256
THEN A=255
60 IF A$=CHR$(17) THEN A=A-1: IF A=-1 THEN
A=0
70 IF A$="M" THEN POKE V+28,255
80 IF A$="N" THEN POKE V+28,0
90 POKE 2040,A
100 GOTO 30
```

Nach dem Programmstart erscheint in der oberen Bildschirmecke die Startadresse des aktuellen Sprite-Bereichs, in der Screen-Mitte sieht man das dazugehörige Sprite-Mu-

Sprite-Register des VIC

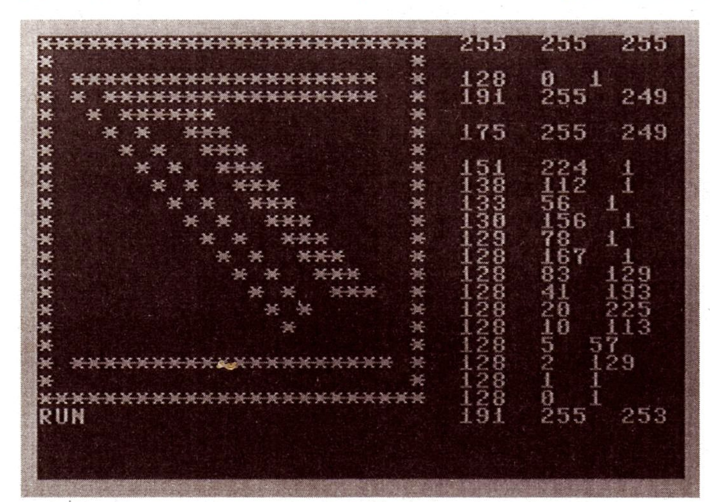
VIC = 53248, NR = Sprite-Nummer (0 bis 7)

Adresse	POKE-Anweisung	Funktion
53248 bis 53263	POKE VIC+2*NR,X	x-Sprite-Koordinate (0 bis 255)
	POKE VIC+2*NR+1,Y	y-Sprite-Koordinate (0 bis 255)
53264	POKE VIC+16,PEEK(VIC+16) OR 2NR	Neuntes Bit zur Steuerung der x-Richtung (muß fürs entsprechende Sprite gesetzt werden, wenn die x-Koordinate während der Sprite-Bewegung größer als »255« wird!)
53269	POKE VIC+21, PEEK(VIC+21) OR 2NR	Sprite einschalten
	POKE VIC+21, PEEK(VIC+21) AND 255-2NR	... ausschalten
53271	POKE VIC+23, PEEK(VIC+23) OR 2NR	Sprite in x-Richtung verdoppeln
53275	POKE VIC+25, PEEK(VIC+25) OR 2NR	Priorität: Sprite erscheint vor dem Bildhintergrund (z.B. Text)
	POKE VIC+25, PEEK(VIC+25) AND 255 - 2NR	... hinter dem Text bzw. der Level-Grafik
53276	POKE VIC+28, PEEK(VIC+28) OR 2NR	Multicolorsprite ein- ...
	POKE VIC+28, PEEK(VIC+28) AND 255 - 2NR	und ausschalten
53277	POKE VIC+29, PEEK(VIC+29) OR 2NR	... in y-Richtung
53278	PEEK(VIC+30): POKE VIC+30,0	Sprite-Sprite-Kollision. Die Bits der beteiligten Sprites sind gesetzt und lassen sich abfragen. Nach dem Zusammenstoß muß man das Kollisionsregister wieder löschen (geschieht nicht automatisch!)
53279	PEEK(VIC+31): POKE VIC+31,0	Sprite-Hintergrund-Kollision (s. Adresse 53278)
53287	POKE VIC+39+NR, Farbcode (0 bis 15)	Sprite-Farbe festlegen

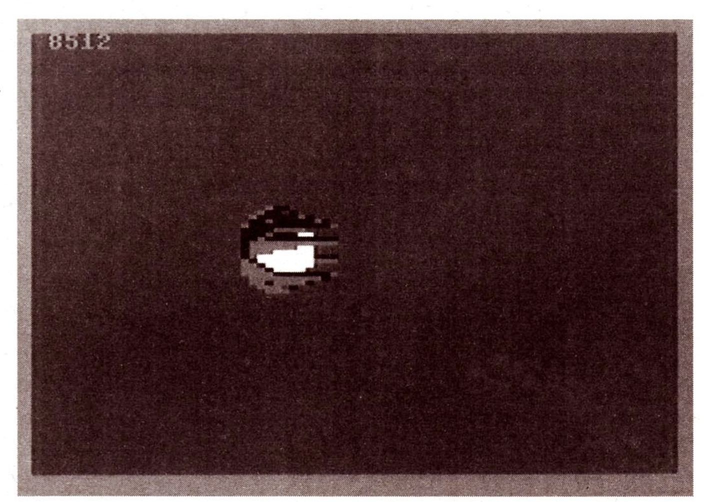
Tabelle 4

Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Dezimalwerte		
Wert / Byte	128	64	32	16	8	4	2	1	128	64	32	16	8	4	2	1	128	64	32	16	8	4	2	1			
0- 2																											
3- 5																											
6- 8																											
9-11																											
12-14																											
15-17																											
18-20																											
21-23																											
24-26																											
27-29																											
30-32																											
33-35																											
36-38																											
39-41																											
42-44																											
45-47																											
48-50																											
51-53																											
54-56																											
57-59																											
60-62																											

[5] Komplettes Entwurfsblatt für Sprites inkl. Erfassung der Dezimalwerte



[6] Mini-Sprite-Editor: Entwurf am Bildschirm



[7] Sprite-View: ausgefiltertes Katakis-Sprite

ster (in x- und y-Richtung vergrößert, Abb. 7). Durch den Speicher blättert man mit <CRSR rechts> und <CRSR abwärts>. Tippt man auf <M>, schaltet man den Multicolorsprite-Modus ein; <N> stellt ihn wieder ab. Maximal 256 Sprites lassen sich im Speicher suchen.

Nachteil: das Mini-Listing enthält keine Speicherroutine, um gefundene Sprite-Muster auf Disk zu sichern. Dazu muß man sich die Startadresse merken und den Bereich z.B. mit einem Maschinensprache-Monitor speichern.

(Chr. Brochhaus/bl)

Bunte Listings

Wer Basic-Listings beim LIST-Befehl auf dem Bildschirm betrachtet, erkennt spezielle Subroutinen und Unterprogramme oft nicht auf Anhieb – schon gar nicht, wenn der Programmierer auf erläuternde REM-Zeilen verzichtet hat. Die haben nämlich einen immensen Nachteil: sie blähen den Umfang eines Programm-Listings unnötig auf.

Mit unserem Trick reicht eine einzige REM-Anweisung, die jedes Unterprogramm einleiten sollte:

```
10 FOR I=49152 TO 49171: READ A: POKE I,A:
NEXT
20 POKE 774,0: POKE 775,192
30 DATA
72,201,143,208,11,200,177,95,201,32
40 DATA 240,3,141,134,2,136,104,76,26,167
```

Nach dem Start mit RUN ergänzen Sie zum Test das Listing:

```
1 REM@
```

Nach der Eingabe von LIST erscheinen die Basic-Zeilen schwarz: der Klammeraffe hat nämlich den Bildschirmcode 0. Diese Zahl wird von der Routine als Farbcode interpretiert. Alle nachfolgenden Listingzeilen erscheinen in dieser Farbe – bis der Interpreter auf eine weitere REM-Anweisung mit einem anderen Zeichen stößt.

Möglich sind auch die Buchstaben »A« (Codewert 1) bis »O« (Codewert 15). Achtung: Höhere Buchstaben (= höherer Codewert) bringen nichts und erzeugen nur Fehlermeldungen.

Arno Gölzer/bl

Es flimmert!

Hier sind zwei interessante Screen-Flimmereffekte zum Ausprobieren. Dazu muß man jeweils die Basic-Zeile 10 eingeben und mit RUN starten:

```
10 FOR I=18 TO 30: POKE 53265,I: NEXT: GO-
TO 10
10 FOR I=1 TO 255: POKE 53270,I: NEXT: GO-
TO 10
```

Gemeinsam laufen die POKEs zu Hochform auf:

```
10 FOR I=18 TO 30: POKE 53270,I: POKE
53265,I: NEXT: GOTO 10
```

Bei dieser Bildschirmspielerei durfte ein weiteres Kontrollregister des VIC-Chip mitmachen: 53265. Bit 3 kümmert sich nämlich um die vertikale Verteilung der Bildschirmzeilen (ob's 24 oder 25 sind).

(Tzimas Kosta/bl)

Text scrollen – ohne Assembler!

Unser Listing verschiebt beliebigen Text Pixel für Pixel von rechts nach links über den Bildschirm. Tippen Sie dazu folgendes Basic-Programm ab, und starten Sie es mit RUN:

```
10 A=53270: L=40
20 A$="{21 SPACES}64'ER-SONDERHEFTE "
30 PRINT CHR$(147)
40 FOR R=1 TO LEN(A$): FOR I=207 TO 200
STEP -1
50 PRINT CHR$(19)MID$(A$,R,L)
60 POKE A,I: NEXT I,R: GOTO 40
```

Die Variable A definiert das VIC-Register 22 (53270), das für horizontales Smooth-Scrolling zuständig ist. L enthält die Anzahl der Zeichen (40 für die gesamte Screen-Breite, weniger für kleinere Textausschnitte). Ist die Zeichenkette kürzer als der Wert L, muß man mit Leerzeichen auffüllen.

Die Stringvariable A$ speichert den auszugebenden Text, dessen letztes Zeichen ebenfalls ein <SPACE> sein sollte: Es wird nämlich nicht gelöscht und bleibt auf dem Bildschirm stehen. Jedes andere Zeichen ergibt eine zeilenlange Schlange auf dem Bildschirm.

Das Programmprinzip: Der Text wird per PRINT auf dem Bildschirm ausgegeben, anschließend der Screen um sieben Pixel nach links verschoben. Der Text rutscht nach, das Scroll-Register wird gelöscht. Zur Textpositionierung auf dem Bildschirm sind alle Cursor-Steuerzeichen erlaubt.

von links nach rechts:

In die andere Richtung geht's mit folgendem Mini-Programm:

```
10 FOR T=0 TO 7: POKE 53270,T
20 FOR G=0 TO 3: NEXT G,T
30 ON A GOTO 10
40 FOR X=1024 TO 2023: POKE X,194: NEXT
50 A=1: GOTO 10
```

Die Scroll-Geschwindigkeit wird in der Schleife (Zeile 20) eingestellt. Der Wert 3 erzeugt minimale Verzögerung. Experimentieren Sie mit höheren Zahlen und beobachten Sie die daraus resultierenden Effekte!

Statuszeilen:

Viele Anwendungen benutzen die ersten zwei oder drei Bildschirmzeilen am oberen Rand als Kopfzeilen (z.B. für Menüs, Programmhinweise usw.). Wenn der Screen vertikal nach oben scrollt, dürfen solche Zeilen nicht verschwinden!

```
10 FOR I=40960 TO 49151: POKE I,PEEK(I):
NEXT
20 FOR I=57344 TO 65535: POKE I,PEEK(i):
NEXT
30 POKE 59639,4: POKE 1,53
```

Zuständig für diesen Trick ist eine ROM-Adresse im Kernel: 59639 ($E8F7), Bestandteil der systemeigenen Scroll-Routine ab $E8EA (59626). Dort steht der Wert $FF (255 = Bildschirmzeile 0), der sich im ROM aber nicht ändern läßt. Deshalb muß man den Basic-Interpreter und das Kernel ins RAM unters ROM kopieren (dauert in Basic fast eine Minute!). Anschließend ändert man den Wert in 59639 (z.B. »4« = vier Kopfzeilen) und das Datenregister 1 auf die RAM-Kopie des Betriebssystems (POKE 1,53). Nach Eingabe von POKE 1,55 ist das ROM wieder aktiv.

(G. Brandt/H. Harmann/P. Eckart/bl)

Serenade für vier Screens und einen Monitor

Nach der Eingabe unseres Listings und anschließendem Start mit RUN kann man ab sofort mit vier Textbildschirmen arbeiten – statt nur mit einem!

```
10 FOR I=828 TO 906: READ A: POKE I,A:
NEXT
20 SYS 828
30 DATA 169,3,141,21,3,169,102,141,20,3,
169,128,141
32 DATA 136,2,169,5,141,24,208,169,1,141,
0,221,169
34 DATA 0,133,51,169,128,133,56,141,132,
2,133,52,96
36 DATA 76,49,234,166,197,224,3,144,247,
224,7,176
38 DATA 243,189,128,3,141,24,208,189,132,
3,141,136
40 DATA 2,24,32,16,229,76,49,234,53,5,21,
37,140,128
42 DATA 132,136
```

Die jeweiligen Screens aktiviert man per Funktionstasten <F1>, <F3>, <F5> und <F7> (falls man nur Byte-Müll erblickt, löscht man den Bildschirmspeicher per <CLR/HOME>!). So kann man z.B auf Screen 1 ein Listing eingeben und sich dazu in Screen 3 oder 4 beliebige Notizen machen, in denen man jederzeit wieder nachschlagen kann.

Für die einzelnen Bildschirme wurde der RAM-Bereich ab $8000 (32768) bis $8FFF (36863) gewählt und die Funktionstastenabfrage in den C-64-Interrupt eingebunden. Ein Tip: Wenn man die Routine per <RUN STOP/RESTORE> verläßt, muß der Zeiger wieder auf den Default-Wert des Bildschirm-RAM eingerichtet werden. Dazu gibt man blind ein: POKE 648,4. Jetzt liegt der Bildschirm wieder in dem Bereich, der nach dem Einschalten des Computers aktiv ist.

(Hauke/bl)

Trickfilm-Wettbewerb

It's Showtime

Sie sind begeisterter Grafiker und Programmierer? Dann kommen Sie gerade recht, denn wir suchen den besten Cartoon-Spot auf dem C 64!

Jetzt heißt es: Sprite-Editor hervorholen und eine gute Idee haben. Gefragt ist der schönste Trickfilm auf dem C 64. Sprites sind wie kein anderes Objekt geeignet, um bewegte Figuren und Objekte auf den Bildschirm zu bringen. Singlecolor-, Multicolor- oder Overlay-Sprites, alles kommt in die Wertung.

Natürlich können Sie Ihre Figur durch eine extra gestaltete Landschaft flitzen lassen. Das Programm, das die Animationen auf den Bildschirm bringt, kann in Basic oder Assembler sein, wichtig für uns sind Idee und Qualität der Animation. Sie können einen Cartoon zeichnen oder auch einen technischen Vorgang (z.B. Otto-Motor) demonstrieren.

1. Preis



Das beste Programm wird mit einem Starfighter-Joystick von Quickshot belohnt (Abb.). Mit ihm können zwei Spieler via Infrarot tolle Stunden am Bildschirm erleben. Platz zwei und drei erhalten je einen Manix-Twin von Dynamics.

Schicken Sie Ihre Animation bis zum 30. November 1993 an:

Markt & Technik Verlag AG
64'er-Sonderheft
Stichwort: Trickfilm
Hans-Pinsel-Str. 2
85540 Haar b. München

Unter allen Einsendern verlosen wir außerdem zehn Sonderhefte freier Wahl. Natürlich ist der Rechtsweg ausgeschlossen.

(lb)

Giga-Publish-Wettbewerb

Olympiade der DTP-Profis

Da sind sie, die Gewinner unseres Layout-Wettbewerbs im 64'er-Sonderheft 88! Ehrlich: wir hatten nicht erwartet, so viele tolle DTP-Seiten zu bekommen ...

Die Qual der Wahl war gewaltig. Leicht haben wir's uns bestimmt nicht gemacht, die Gewinner aus der Flut der Einsendungen herauszufiltern. Die Bewertungskriterien waren vor allem optische Seitenaufteilung, der Einsatz von Grafik bzw. unterschiedlichen Zeichensätzen, aber auch die Idee (= Verwendungszweck), die den DTP-Seiten als Grundlage diente und möglichst viele Leser anregen soll, ähnliches mit Giga-Publish (oder vergleichbaren C-64-DTP-Programmen) zu fabrizieren.

Aufgrund der unerwartet hohen Beteiligung am Wettbewerb haben wir uns spontan entschlossen, das Siegertreppchen (s. Tabelle) auf fünf Stufen aufzustocken. Die Ergebnisse können Sie in Text und Bild auf dieser und den folgenden Seiten bewundern.

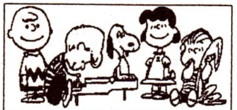
4. Platz: Stellenanzeige – das mit dem Häuschen im Grünen ist allerdings schamlos übertrieben!

Haben Sie Ihren alten Job satt?

Dann kommen Sie zu uns!



Wir brauchen Verstärkung in der Redaktion



Wir sind der führende Fachverlag für Computerwissen mit über 350 Mitarbeitern und mehreren Tochtergesellschaften im In- und Ausland. Für Fragen und eine erste Kontaktaufnahme steht Ihnen Chefredakteur Georg Klinge (auf dem Bild l.v.l.) zur Verfügung. Tel.: 089/4613169

Unsere Anforderungen

Gute Kenntnisse des PC und des C 64, die Fähigkeit, sich klar und verständlich auszudrücken, Bereitschaft zur Teamarbeit und Kreativität sind eine wichtige Vorraussetzung. Eine Berufserfahrung als Redakteur ist nicht unbedingt erforderlich. Selbstverständlich werden Sie gründlich eingearbeitet.

So muskulös, wie der junge Mann auf dem Foto rechts, müssen Sie nicht aussehen.





Unser Angebot:

Wir bieten ein ausgezeichnetes Betriebsklima, zukunftssichere Arbeit in einem jungen, dynamischen Team, leistungsgerechte Bezahlung, gute Sozialleistungen, eine betriebliche Altersversorgung und vieles mehr.

Haben wir Ihr Interesse geweckt?

Dann rufen Sie heute noch an!

Vielleicht können auch Sie sich bald ein Häuschen im Grünen leisten.

Machen Sie Ihr Hobby zum Beruf!

Vielen Dank an alle, die mitgemacht haben! Und wenn's diesmal nicht geklappt hat: neue Chance, neues Glück – auch bei unserer nächsten Konkurrenz – Mitmach-Aufruf in diesem Sonderheft (»Der beste Trickfilm«) – gibt's tolle Preise zu gewinnen!

5. Platz: Cluburkunde – raffinierte Grafiksymbole

Neptun und Aiolos haben heute, im Jahr der olympischen Spiele

Riccardo Biffi

nach alter Väter Sitte in die Gemeinschaft der Yacht-Club See- und Fahrensleute aufgenommen.



Möge Dir Kameradschaft und Seemannschaft immer gegenwärtig sein!

Mast- und Schotbruch!



Yacht-Club Luzern
Der Präsident:



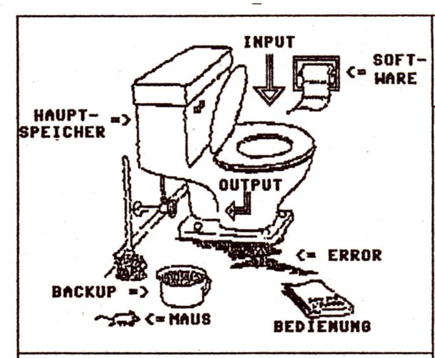


22. Mai 1998

Die Sieger

Plaz.	Name, Wohnort	Preis
1	Karsten Loffeld, Krefeld-Fischeln	Scanntronik-Handyscanner
2	Frank Rietz, Hettstedt	Book-Ware inkl. Disketten
3	Olaf von Paul, Leutesdorf	fünf 64'er-Sonderhefte
4	Frank Engel, Saarlouis-Steinrausch	drei 64'er-Sonderhefte
5	Rosemarie Fridrich, CH-Ennetburgen	zwei 64'er-Sonderhefte

1. Platz: KLO-Report – der Listing-Klau geht um!



KLO ~ Report

Unauffindbare Listings !

Wer hat Sie geklaut ?
Wer kann uns helfen Sie zu bekommen ?

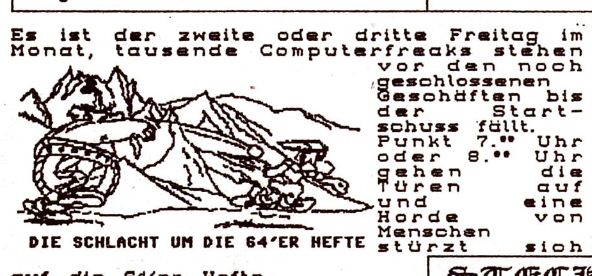
Leser wissen sich keinen Rat mehr !
Ob die 64'er Redaktion dahinter steckt ?

Mehr dazu in unserem Bericht !



KLO am Tag
by Karsten LOffeld

Es ist der zweite oder dritte Freitag im Monat, tausende Computerfreaks stehen vor den noch geschlossenen Geschäften bis der Start-schuss fällt. Punkt 7.°° Uhr oder 8.°° Uhr gehen die Türen auf und eine Horde von Menschen stürzt sich auf die 64'er Hefte.


DIE SCHLACHT UM DIE 64'ER HEFTE

Nach einer großen aber gewonnenen Schlacht durch die Menge hat man dann das begehrte Heft in seinen Händen und geht vergnügt nach Hause. Kaum zu Hause angekommen (völlig im Compifieber) beginnt man dann eifrig zu lesen.
Nach so etwa zwanzig bis dreizig Seiten kommt dann ein Superlisting, das die 64'er Redaktion als "Listing des Monats" heraus gibt.
Dieses Listing springt einen förmlich mitten ins Gesicht und man beginnt eifrig sich die Bedienung des Programms einzuprägen. Am Ende des Artikels wird man dann, gerade erst vom Listing begeistert, aus seinen schönsten Träumen gerissen.

Wo ist das Listing ?
Man kann noch solange hin und herblättern wie man will ...., nichts !
Der Listingsklau war wieder da !
Aber Moment mal, da war doch noch was !
Ja, jetzt sieht man etwas. Ganz unten in der Ecke steht eine kleine Nachricht, wie man das Listing bekommen kann.

Da hat man die ganze Bedienung des Programms gelesen, aber kein Programm. Das ist nicht fair !
Also was muß man machen ? Ach ja, da steht es ja nochmal ! Man kann dieses und die anderen Programme auf einer sogenannten Servicediskette bekommen, die dann DM 9.80 kostet. Das macht schon mit den DM 7,80 vom Heft satte DM 17,60. Die zweite Möglichkeit wäre BTX (wer hat bzw. kann sich schon BTX leisten). Und die dritte, einen an sich adressierten Rückumschlag DM 2,40 jetzt (DM 3,00) wo wir dann schon bei DM 10,20 bzw (DM 10,80) wären und noch die ganze Arbeit mit dem Abtippen.

AB DAMIT ZUR POST

Was bleibt ist die letzte und die billigere Lösung, den Rückumschlag von DM 2,40.
Voller neuer Hoffnung, das Listing endlich in den Computer zubekommen, schreibt man an die Redaktion. Die das Listing, auch wenn es etwas dauert (da man ja bekanntlich nicht allein auf der Welt ist) verschickt.
Jeden Morgen flitzt man, gierig zum Briefkasten und schaut nach, ob Es schon gekommen ist. Und dann eines schönen Morgens ist es da, aber wie ???
Man weiß ja, wie vorsichtig die Post unsere Briefsendungen behandelt (ironisch gemeint). Der Rückumschlag sieht aus als hätte man Ihn durch die Mangel gedreht. Total zerknittert, leicht eingrissen, aber wie durch ein Wunder ist der Inhalt Gott sei Dank nicht beschädigt worden.
Jetzt hat man das so sehr gewünschte Programmlisting in den Händen, auch wenn es einige Tage langen Zitterns gedauert hat.

STECKBRIEF

Geburtsdatum : Heft Nr. 01/92
Größe : unbekannt da er nur Seiten klaut

Phantombild : siehe Bild
Fingerabdrücke : leider keine
Straftat : das Klauen von Listings

Vorliebe : Listings des Monates
Merkmale : hinterläßt eine kleine Nachricht

Ende der Story, über einen Alltag der 64'er Leser.

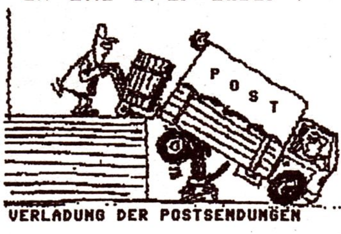
Aber muss das denn sein, liebe 64'er Redaktion ??

Unser KLO-Reporter hat sich dann, mit vollem Eifer auf die Suche nach dem geheimnisvollen Klauer gemacht. Er fegte wie ein Wirbelsturm durch die Heftseiten um <u>IHN</u> zufinden. Was allerdings nicht gerade einfach war, da man ab Heft Nr. 04/84 anfangen mußte. Die Jahre (84/85/86/87/88/98/90/91) vergingen wie im Fluge, aber keine Spuren des vermeintlichen Klauers in Sicht.
Doch da, Heft Nr. 01 von 1992, er hatte zugeschlagen, zum ersten aber leider nicht zum letzten Mal. Denn seit diesem Heft treibt er sein Unwesen in unseren Heften.

Jeder macht mal Fehler und fliegt auf, nicht wahr, liebe 64'er Redaktion !

Daher ein Aufruf an alle 64'er Leser !

Wir bitten um Eure Mithilfe, damit wir dem Listingsklauer schnellstens fassen, oder wenigsten seine Arbeit einschränken können.



VERLADUNG DER POSTSENDUNGEN

Für Hinweise, die zur überführung des Täters führen, wenden Sie sich an alle Computerfreaks oder an den KLO-REPORT direkt !
Diesen Hinweisen werden wir dann sofort nachgehen, und für eine Regelung sorgen, die allen Lesern gerecht wird.

Text: Karsten Loffeld
Fotos: KLO-REPORT

PS. Ein dankeschön an alle, die mir bei diesem Projekt geholfen haben !

Deutschland in Stichworten

Geographische Lage: Mitteleuropa, gemeinsame Grenzen mit Dänemark, den Niederlanden, Belgien, Luxemburg, Frankreich, der Schweiz, Österreich, der Tschechischen Föderativen Republik, Polen

Fläche: 356 957 km²

Einwohner: 79,1 Millionen (davon 41,0 Millionen Frauen, 38,1 Millionen Männer; über fünf Millionen Ausländer), Bevölkerungsdichte: 222 Menschen je km²

Verfassung: Grundgesetz für die Bundesrepublik Deutschland vom 23. Mai 1949

Staatsform: Bundesstaat, gegliedert in 16 Länder. Mit dem Wirksamwerden des Beitritts der DDR (Beschluß der Volkskammer vom 23. August 1990) zur Bundesrepublik Deutschland am 3.Oktober 1990 wurden die Länder Mecklenburg-Vorpommern, Brandenburg, Sachsen-Anhalt, Thüringen und Sachsen errichtet und Glieder der Bundesrepublik Deutschland. Die elf Bezirke von Berlin-Ost wurden mit dem Land Berlin vereinigt.

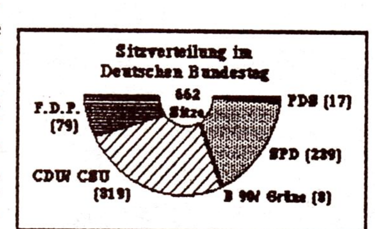
Volksvertretung: Deutscher Bundestag, am 2. Dezember 1990 in freier, gleicher, allgemeiner und direkter Wahl zum ersten Mal gesamtdeutsch gewählt. Die Wahlperiode beträgt vier Jahre. Eines der fünf Mitglieder des Präsidiums des Deutschen Bundestages leitet jeweils die Plenarsitzungen. Der größte Teil der parlamentarischen Arbeit wird in den Ausschüssen geleistet, die sich jeweils mit einem bestimmten Sachgebiet beschäftigen.

Berlin
Hamburg
München
Köln
Frankfurt/M.
Essen
Dortmund
Düsseldorf
Stuttgart
Bremen
Duisburg
Leipzig

Deutschlands größte Städte

Staatsoberhaupt: Der für fünf Jahre von der Bundesversammlung gewählte Bundespräsident. Die Bundesversammlung setzt sich zur Hälfte aus den Mitgliedern, die von den Volksvertretungen der Länder gewählt werden, zusammen. Der Bundespräsident vertritt die Bundesrepublik Deutschland völkerrechtlich. Zu seinen Aufgaben gehört auch die Ernennung des vom Deutschen Bundestag gewählten Bundeskanzlers und der Bundesminister. Bundespräsident ist zur Zeit Dr. Richard von Weizsäcker (1989 zum zweiten Mal gewählt).

Bundesregierung: Die Bundesregierung besteht aus dem Bundeskanzler und den Bundesministern. Der Bundeskanzler bestimmt die Richtlinien der Politik. Innerhalb dieser Richtlinien leiten die Bundesminister ihre Ressorts in eigener Verantwortung.



Bundeskanzler ist seit 1982 Dr. Helmut Kohl (CDU, nach dem Wahlsieg der Koalitionsparteien CDU, CSU und F.D.P. vom 2. Dezember 1990 zum dritten Mal im Amt bestätigt)

Nationalflagge: Schwarz-Rot-Gold

Für die Herkunft der Flaggenfarben gibt es mehrere Erklärungen. Eine besagt, daß sie auf die Uniformen eines Freikorps zurückgehen, das im frühen 19. Jahrhundert gegen die napoleonische Besatzung in Deutschland kämpfte. Zu Nationalfarben wurden sie erstmalig durch die in Frankfurt am Main tagende Nationalversammlung, die aus der - später scheiternden - deutschen Revolution von 1849/49 hervorgegangen war.

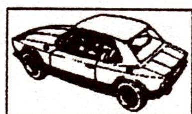
Nationalfeiertag: 3. Oktober, Tag der Deutschen Einheit

Nationalhymne: „Das Lied der Deutschen", dessen dritte Strophe, beginnend mit den Worten „Einigkeit und Recht und Freiheit", gesungen wird. Der Text stammt von August Heinrich Hoffmann von Fallersleben (1798-1874). Die Melodie (zu einem anderen Text komponiert) ist von Joseph Haydn (1732-1809).

Währung: Deutsche Mark (DM, 1 DM = 100 Pfennig) Währungs- und Notenbank ist die von Weisungen der Bundesregierung unabhängige Deutsche Bundesbank (Sitz in Frankfurt am Main). Die Bundesbank ist „Hüter der Währung", d.h. der Sicherung der Geldwertstabilität

Infrastruktur: 10 571 km Autobahnen, 44 275 km Bundes- und Staatsstraßen, über 80 000 km Bundesbahn-Straßen. Die wichtigsten Flughäfen (in alphabetischer Reihenfolge): Berlin-Schönefeld, Berlin-Tegel, Bremen, Dresden, Düsseldorf, Frankfurt am Main, Hamburg, Hannover, Köln-Bonn, Leipzig, München, Nürnberg, Stuttgart).
Über 6 500 km Binnenwasserstraßen.
Größter Binnenhafen: Duisburg.
Größte Seehäfen: Hamburg und Bremen/Bremerhaven an der Nordsee; Kiel; Lübeck und Rostock an der Ostsee.

Wichtigste Industriezweige: Automobilindustrie (Bild unten), Maschinenbau, Elektrotechnik, chemische und pharmazeutische Industrie, Eisen und Stahlindustrie



3. Platz: die Filmklappe – aus dem Alltag eines imaginären Fernsehstudios



Die Filmklappe



Chris Tian zu Besuch bei Studio S

vp. Am 30.06.93 wird der Illusionist Chris Tian bei uns zu Besuch sein

Vorgesehen ist ein Interview mit ihm, sowie die Vorstellung seines neuesten Programms "Welcome to my magic.".

In diesem Programm werden Illusionen gezeigt, wie die Erscheinung mehrer Gegenstände, die "schwebende Kugel", u. a. M. Tian stellt sogar die Vorführung eines Kunststückes in Aussicht, welches bisher noch nicht in seinem Programm zu sehen war. Ob dies jedoch bis zum o.g. Termin möglich sein wird, war bis Redaktionsschluss noch nicht bekannt.



Symbol der Magie: der Zylinder

Auf jeden Fall, so M. Tians Manager, werde Studio S die Möglichkeit erhalten, die gesamte Show aufzuzeichnen. Wann dies genau sein wird, werden wir in einer der nächsten Ausgaben unserer Zeitung oder unserer Sendung bekanntgeben.

Studio S präsentiert die 1. Ausgabe ihrer Zeitung

Dies ist die erste Ausgabe unserer Zeitung.

Einen "eigenen" Sender hatten wir ja schon immer und nun soll dies die Zeitschrift dazu sein.

Wir hoffen, dass wir euren Geschmack getroffen haben. Solltet ihr jedoch Anregungen, Kritik, Lob oder sonst etwas habe, so findet ihr unsere Adresse im Imprssum (was immer das auch sein mag).

Geplant sind bei uns folgende Rubriken:
Who is who in Studio S
Bernado Canneloni
Erich Berger
Andernacher Song Contest



Unser Zeitungslogo: Die Klappe

Eine kleine Bemerkung noch zur Reschtschreipunk: wir sind ser bemüt Veler zu fermeiden, toch glapt dies nicht imer, besonters dan nicht wbn der sezer bedsofen issssssst. alkjdflkweoildküü***

Studio S im offenen Kanal Andernach

vp. Schon seit längerer Zeit ist es geplant. Studio S soll im offenen Kanal Andernach auf Sendung gehen.

Jedoch gibt es noch einige Schwierigkeiten. So sind sich die Macher noch nicht einig, welche Beiträge gesendet werden sollen. Auch die Vorgabe des offenen Kanals, dass keine Werbung gesendet werden darf, macht das gesamte Projekt nicht sehr einfach. Was tun sprach Zeus???



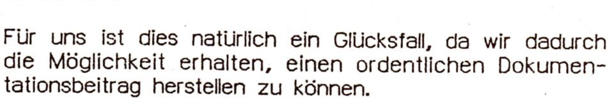
Es bleibt zur Zeit nichts anderes übrig, als sich erneut hinzusetzen und sich alle Beiträge anzusehen. Jedoch bleibt das an Michael und Olaf hängen, da Sacha ja nun die Medien in Amerika studieren will.

Also liebe Leser: Dranbleiben und abwarten.

Sacha Schneider in Amerika
Auf in den Wilden Westen

Jeder, der unser Programm schon einmal gesehen hat, kennt den charmanten jungen Mann mit den langen Haaren: Sacha Schneider.

Unser Freund und Mitredakteur wird nun jedoch eine Weile von unserem Bildschirm verschwinden, da er nämlich für einen Monat die Aussenkorrespondenz in den USA übernimmt.



Für uns ist dies natürlich ein Glücksfall, da wir dadurch die Möglichkeit erhalten, einen ordentlichen Dokumentationsbeitrag herstellen zu können.

Was unseren Freund drüben alles erwartete und was sich dort ereignete, dies werden wir in unserem Programm und in unserer Zeitung dann ausführlich erläutern.

Bis dahin wünschen wir Sacha einen guten Flug, viel Spass und eine gute Heimreise.

IMPRESSUM

"Die Filmklappe ist die Begleitzeitschrift zu den Sendungen von Studio S.

Sie erscheint unregelmässig.
Chefredaktion: Olaf von Paul
Red. Erich Berger: Sacha Schneider
Red. Who is who: Michael Strubel
Anregung, Kritik, Lob sind zu richten an:
Olaf v. Paul, Vordergasse 6, 5458 Leutesdorf

Turtle-Grafik – rasant und komfortabel

Der Pixel-Hexer

Nur Experten wissen, welche fantastischen mathematischen Möglichkeiten man mit der Programmiersprache »Logo« und der darin integrierten »Turtle-Grafik« hat. Unser Programm simuliert diese Befehle auf dem C 64, wie's Logo selber nicht besser können!

Oft wird sie als Anfängersprache abgetan: Logo und die »Turtle-Grafik«, hierzulande auch als »Igel-Grafik« bekannt. Darunter versteht man eine Reihe von Befehlen, mit denen man eine gedachte Schildkröte (engl. turtle) über den Bildschirm bewegt.

Vollkommen unüblich wird ein Punkt auf der Bitmap nicht durch ein Koordinatenpaar bestimmt (obwohl das möglich wäre), sondern man sagt der Schildkröte, daß sie sich um einen bestimmten Winkel nach links oder rechts drehen und dann eine gewisse Anzahl Punkte vorwärts oder rückwärts bewegen soll. Mit diesem simplen System erzeugt man anspruchsvolle Grafiken, für die der normale Rechenaufwand ungleich größer wäre. Vor allem durch Iteration und Rekursion lassen sich sehr komplexe mathematische Bilder mit kurzen Basic-Programmen herstellen.

Obwohl Turtle-Grafik aus Geschwindigkeitsgründen in optimierter Maschinensprache programmiert wurde, braucht man zur Anwendung nur grundlegende Basic-Kenntnisse. Die neuen Befehle werden durch das Anführungszeichen <!> eingeleitet und lassen sich wie normale Basic-Befehle einsetzen. Die Basic-Erweiterung fängt fast alle Bedienungsfehler ab. Fertige Grafiken lassen sich auf Diskette speichern und später mit einem Hardcopy-Programm drucken – Turtle-Grafik besitzt keine entsprechenden Routinen. Das Programm benutzt den hochauflösenden Modus des C 64 (320 x 200 Punkte).

Laden Sie die Basic-Erweiterung mit:

```
LOAD "TURTLE-GRAFIK",8
```

Gestartet wird mit RUN.

Der Bildschirm bringt die Einschaltmeldung: ab sofort lassen sich die neuen Befehle und Funktionen aufrufen. Wer ganz sicher gehen will, sollte die Grenze des freien Basic-RAM auf Adresse $8C00 (35840) heruntersetzen (damit keine selbstfabrizierten Basic-Programme und deren Variablen mit dem Speicherbereich kollidieren, den Turtle-Grafik benutzt):

```
POKE 56,140 : CLR
```

Hier die Übersicht aller neuen Anweisungen (in alphabetischer Reihenfolge):

!ANGLE W

... bestimmt die Richtung. Der Parameter »W« ist die Gradzahl des vorgesehenen Winkels (0 bis 359):

0 zeigt nach oben, 90 nach links, 180 nach unten und 270 nach rechts. Winkel über 359 Grad korrigiert das Programm automatisch. Vermeiden Sie aber Werte über ca. 32000 Grad! Die Position der Turtle ändert sich nicht, gezeichnet wird ebenfalls nichts.

TURTLE

SO SIEHT **FETTSCHRIFT** AUS.

- FETTSCHRIFT
READY.

[1] Turtle-Schrift: geschickte Programmierung ersetzt neue Zeichensätze

!BACK N

Die Schildkröte bewegt sich von ihrer aktuellen Position um N Schritte rückwärts. Ob dabei gezeichnet wird, bestimmen der !PEN bzw. !MODE-Befehl (s. Beschreibung). N liegt im Bereich von 0 bis 65535. Achten Sie darauf, daß die Turtle nicht den Bildschirm verläßt! Je nach Einstellung von »!RAY« (s. Beschreibung) wird die neue Turtle-Position nach dieser Anweisung gespeichert oder nicht. !BACK hat die gleiche Wirkung wie die Drehung um 180 Grad vor einem entsprechenden !WALK-Befehl, dem ein erneuter Dreh um 180 Grad folgt.

Beispiel: !BACK 80

!CLEAR

... löscht den Grafikbildschirm. Weder die Turtle-Stellung noch der Anzeigemodus ändern sich.

!DISCARD "Name",8

... hat die gleiche Funktion wie SAVE. Es wird die im Speicher vorhandene Grafik im Hi-Eddi-Standard-Format (32 Blocks) auf Diskette gesichert. Das Bild läßt sich mit !FETCH oder !MERGE wieder laden (s. Beschreibung).

Beispiel: !DISCARD "TESTBILD",8

!FETCH "Name",8

... funktioniert wie der LOAD-Befehl. Die Grafik darf nicht größer als 33 Blocks sein, also ohne Farb-RAM. Das aktuelle Bild im Grafikspeicher wird gelöscht.

Beispiel: !FETCH "TESTBILD",8

!FINISH

... sollte am Ende Ihrer Turtle-Grafikprogramme stehen: die Anweisung aktiviert den Grafikmodus, löscht den Tastaturpuffer, wartet auf einen beliebigen Tastendruck, wechselt in den Textmodus und beendet das Basic-Programm.

!GRAPHICS

... schaltet den Grafikbildschirm ein. Erst dann sieht man, was die Schildkröte zeichnet – obwohl auch im Textmodus alle Zeichenbefehle quasi »unsichtbar« ausgeführt werden! Ist !WINDOW (s. Beschreibung) aktiviert, ändert sich nach der !GRAPHICS-Anweisung nichts.

!HEADING

... gibt Auskunft über den aktuellen Winkelgrad der Turtle (s. !ANGLE). Nützlich für anschließende Berechnungen.

Beispiel: PRINT !HEADING

!HOME

Die Schildkröte bewegt sich exakt zur Bildschirmmitte, ohne eine Linie zu zeichnen (x = 160, y = 100, Winkel = 0).

!INIT

... versetzt die Basic-Erweiterung in den Normalzustand: Grafikbildschirm löschen, !HOME-Befehl ausführen, Stift aufs »Papier« setzen (PEN DOWN) und Funktion »Punkte setzen« einstellen (!MODE 1 bzw. !PEN 1). Außerdem stellt das Programm die !RAY-Option ab und definiert das Linienmu-

ster mit dem Wert 255 (alle Punkte). Der Stack wird gelöscht, der Window-Modus abgeschaltet und die Grafikfarbe auf weißen Hintergrund mit schwarzen Punkten getrimmt. Der Computer führt !INIT automatisch beim Start von Turtle-Grafik oder nach der Anweisung »SYS 49152« aus. Es ist aber empfehlenswert, diesen Befehl zu Beginn eigener Turtle-Programme zu verwenden.

!INVERS

... invertiert den Grafikbildschirm. Weder Turtle-Position noch Anzeigemodus ändern sich.

!KARTX R,W

... rechnet Polarkoordinaten in karthesische Koordinaten um und liefert die x-Komponente. R ist der Radius, W der Winkel in Grad (zwischen 0 und 359). Als Basis dient diese Formel: KARTX R,W = R * SIN (W/180 * pi), mit 4/5-Rundung.

Beispiel: !MOVE !KARTX 17,90,4

!KARTY R,W

... wie !KARTX, aber für die y-Komponente (vertikal). Die Formel: KARTY R,W = R * COS (W/180 * pi)

!KILL

... schaltet die Turtle-Grafik ab: Basic 2.0 übernimmt wieder die Vorherrschaft. Verwenden Sie diesen Befehl, wenn Sie andere Erweiterungen laden möchten. Sofern sich Turtle-Grafik noch im Speicher befindet, kann man das Programm mit SYS 49152 wiederbeleben (!INIT wird automatisch ausgeführt!).

!MERGE "Dateiname",8

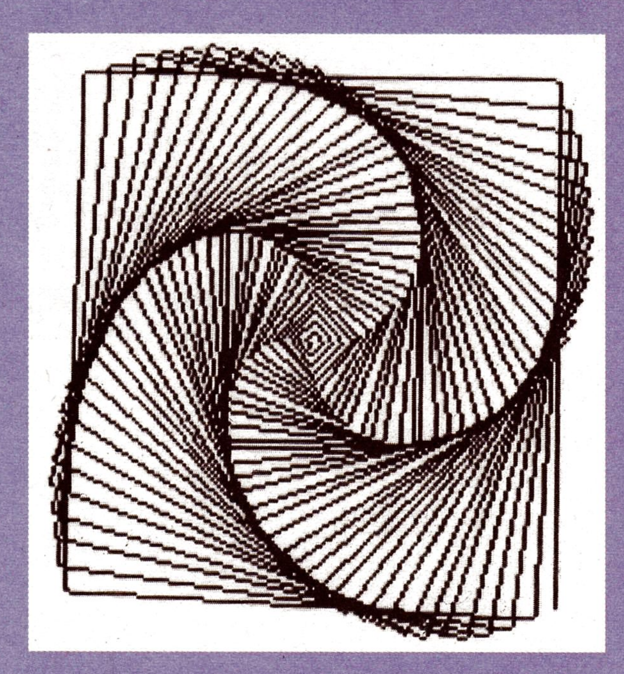
... holt wie !FETCH eine Grafik von Diskette, läßt die alte aber im Speicher und kopiert die neue darüber. Der Stack wird mit !RESUME (s. Beschreibung) gelöscht.

Beispiel: !MERGE "TESTBILD",8

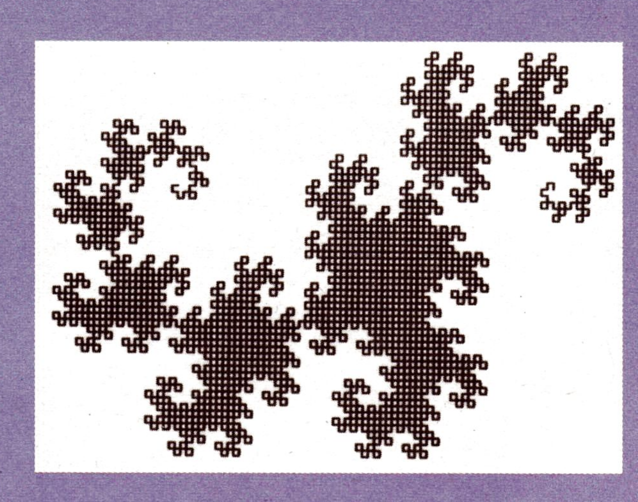
!MODE M

... bestimmt den Turtle-Zeichenmodus:

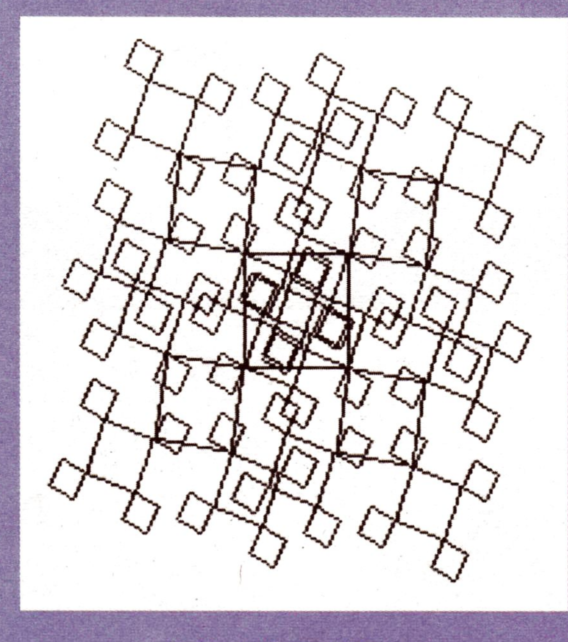
– M = 0: zeichnen und Punkte löschen,
– M = 1: zeichnen und Pixel setzen (Normaleinstellung).
– M = 2: zeichnen und Punkte invertieren
– M = 3: wie PEN UP, die Turtle zeichnet nicht!



[2] Turtle-Demo 2: Viereck



[3] Turtle-Demo 1: Drachen



[4] Turtle-Demo 3: Blume

Nur die Bits 0 und 1 des Parameters M werden ausgewertet.

Beispiele: !MODE (0.25*4), !MODE 1

!MODE

... gilt ohne Parameter als Funktion. Sie liefert in den Bits 0 und 1 den per !MODE M-Befehl eingestellten Zeichenmodus (3 = PEN UP). Bit 2 ist bei aktiviertem !RAY-Modus gesetzt, Bit 3 bei eingeschalteter Grafik.

Beispiel: PRINT (!MODE AND 8)

!MOVE X,Y

Die Turtle bewegt sich vom augenblicklichen Standort zu den x/y-Koordinaten und zieht dabei eine Linie, falls der Stift aufliegt (s. !PEN). Dabei gilt das normale Hires-Koordinatensystem des C 64. Die Richtung (Winkel) bleibt unverändert. Bei angehobenem Stift bleibt die !RAY-Anweisung wirkungslos. Getrennt durch ein Komma, dürfen hinter !MOVE zwei beliebige numerische Ausdrücke stehen (Zahlen, Variablen, Rechenzeichen).

Beispiel: !MOVE 10*X,20+PEEK(146)

!PATTERN P

Der Parameter »P« ist das binäre Linien-Muster (acht Bit, wahlweise ein- oder ausgeschaltet). Damit legt man fest, daß die Turtle bei Zeichenbefehlen keine durchgehende Linie zeichnet (was beim Maximalwert P = 255 der Fall wäre), sondern z.B. nur jeden zweiten Punkt (P = 85), jeden vierten (P = 136), oder Strichpunkte (P = 228).

Beispiel: !PATTERN 128+64+8+4

!PEN X

Ähnlich wie bei !MODE stellt man hier ein, ob die Schildkröte zeichnet oder sich nur bewegt. Bei X = 0 ist der PEN UP-Modus (!MODE 3) aktiv: nicht zeichnen. Bei anderen Werten (z.B. X = 1) setzt die Turtle Punkte (PEN DOWN, !MODE 1). Nach Einschalten des PEN-DOWN-Modus (gilt nicht für !MODE) erscheint an der aktuellen Position ein Grafikpixel.

Beispiel: !PEN 1 (animiert die Turtle zum Zeichnen)

!POLY A,N,W

Jetzt geht's »A x N« Schrit-

te vorwärts; dabei dreht sich die Schildkröte jeweils um W Grad nach links. Erst sind's nur N/2 Schritte (abgerundet), beim letzten Mal ebenfalls (aufgerundet). Damit kann man komfortabel Polygone, N-Ecke, Sterne usw. auf den Screen bringen. Hier gilt die Formel (die das Programm aber aus technischen Gründen intern nicht selbst berechnen kann): W = 360 / (A-1).

Beispiel: !POLY 6,30,72 (Fünfeck)

!PULL A [,A]

... holt numerische Realvariablen vom Stack, die dort vorher mit !PUSH (s. Beschreibung) gespeichert wurden.

Beachten Sie aber, daß der Stack LIFO-organisiert ist (last in, first out). Die Befehle !PUSH A,B:!PULL A,B vertauschen also z.B. die Inhalte der Variablen A und B. Hinter !PULL darf man beliebig viele Variablennamen angeben, getrennt durch Kommas.

Beispiel: !PULL TY,TX

!PUSH B [,B]

... legt die hinter !PUSH angegebenen numerischen Variablen auf einen programminternen Stack, der maximal 1637 Einträge faßt. Hinter !PUSH kann man (wie bei !PULL) ebenfalls beliebig viele numerische Ausdrücke (Zahlen, Variablen, Terme) eintragen. Dieser Stack eignet sich ideal für rekursive Programme (in denen sich bestimmte Subroutinen selbst aufrufen). Häufig muß man Variableninhalte vor dem Aufruf retten und nach Abarbeitung der Routine mit !PULL wieder zu holen.

Beispiel: I=15:!PUSH I, !PUSH 15

!RAY R

... aktiviert eine spezielle Betriebsart des Programms (X = 1) oder stellt sie ab (X = 0): Nach jedem !WALK, !BACK oder !MOVE-Befehl wird die Turtle wie am Gummiband wieder zur ursprünglichen Position zurückgezogen. Damit kann man z.B. Strahlen und Sterne zeichnen. Bei aktivem PEN-UP ist der Modus allerdings unwirksam.

Beispiel: !RAY 0

!RESUME

... löscht den von !PULL und !PUSH verwendeten Stack.

!SCREEN F

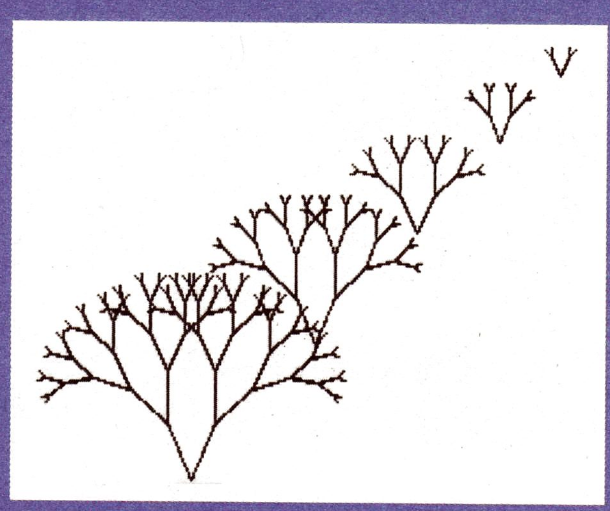
... kontrolliert die Grafikfarben. Den Farbcode F (Normalwert: 1) errechnet man nach der Formel »F = Punktfarbe * 16 + Hintergrundfarbe«. Zur Sicherheit sollten Sie nach jedem !SCREEN-Befehl bei aktiviertem WINDOW-Modus die !GRAPHICS-Anweisung absetzen!

Beispiel: !SCREEN 0*16 + 5

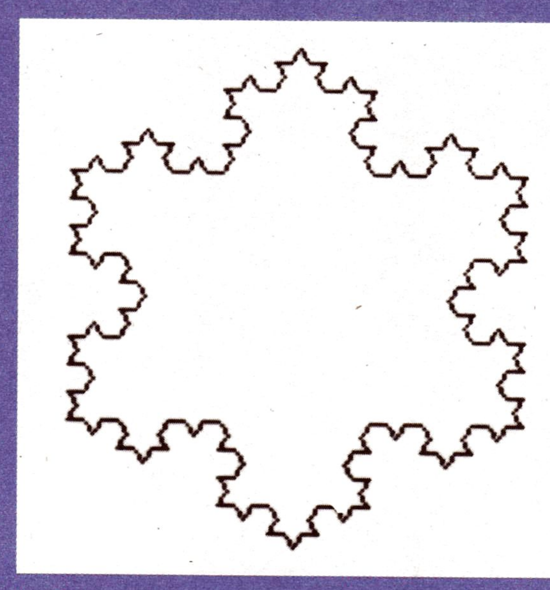
!STACK

... bringt den Wert des Stackzeigers für !PULL und !PUSH (die Anzahl der schon im Stack gespeicherten Elemente). Hier gilt die Formel: »aktuelle Speicherposition des Stack-Endes = !STACK * 5 + 57344« (eine Einheit entspricht fünf Bytes).

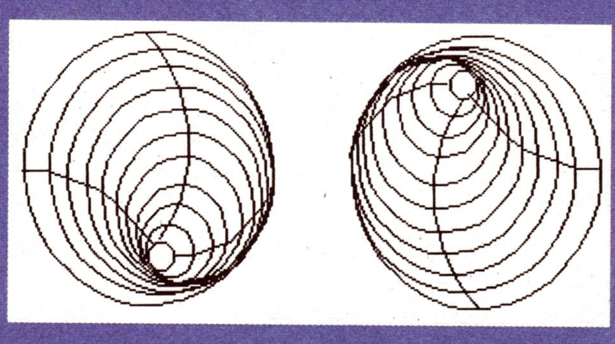
Beispiel: IF !STACK = 0 THEN PRINT "STACK LEER!"



[5] Turtle-Demo 4: Allee



[6] Turtle-Demo 5: Schnee



[7] Turtle-Demo 6: Tunnel

!TEXT

... deaktiviert den Grafikbildschirm und schaltet den Textmodus wieder ein. Hier arbeiten alle Turtle-Befehle versteckt: das Ergebnis kann man erst nach !GRAPHICS oder !WINDOW-Befehl betrachten!

!TURNL W

... dreht die Turtle um W Grad entgegen des Uhrzeigersinns um ihre eigene Achse (also nach links) und bestimmt so deren neue Lage. Vom Positionswinkel zieht man den Wert W ab. Winkel unter 0 Grad korrigiert das Programm automatisch. W ist eine Integerzahl oder ein numerischer Ausdruck zwischen 0 und 65535, wobei man Drehwinkel über 32 000 Grad vermeiden sollte. Die Position der Turtle wird nicht verändert und nicht gezeichnet.

Beispiel: !TURNL 180 (kehrt die Bewegungsrichtung um)

!TURNR W

... wie !TURNL, diesmal geht's aber nach rechts. Zum Positionswinkel der Turtle wird W also addiert. Winkel über 359° korrigiert das Programm selbständig.

Beispiel: !TURNR WI*2

!TURTX

... bringt die augenblickliche x-Koordinate der Turtle im Bereich von 0 (links) bis 319 (rechts); geeignet für Berechnungen.

Beispiel: !WALK !TURTX-15

!TURTY

... ergibt die aktuelle y-Koordinate der Schildkröte (Bereich = 0 oben bis 199 unten).

Beispiel: D=4*!TURTY

!WALK N

Mit dieser Anweisung bewegt man die Turtle von ihrer aktuellen Position um N Schritte in die aktuelle Richtung. Ob gezeichnet wird, legen die Befehle !PEN bzw. !MODE fest. N kann Werte von 0 bis 65535 annehmen. Achten Sie darauf, daß die

Kröte nicht den Bildschirm verläßt! Je nach Einstellung von !RAY-Befehl wird die neue Turtle-Position gespeichert oder nicht.

Beispiel: !WALK K*4

!WINDOW D

... erlaubt Text und Grafik auf ein und demselben Bildschirm. D ist die Bildschirmrasterzeile, bei der man die Fenster (Hires/Lores) trennen will (normalerweise zwischen 50 und 248). Beachten Sie, daß aktivierte Windows oft flimmern, vor allem beim Zeichnen. Schalten Sie daher das Fenster nur am Ende eines Programms, im Direktmodus oder beim Test ein. Ist D = 0 oder 1, schaltet sich diese Betriebsart ab: Text (D=0) bzw. Grafik (D=1) ist wieder aktiv.

Beispiel: !WINDOW 217

Infos zum Programm

Wie bei vielen anderen Basic-Erweiterungen des C 64 gilt auch hier die Einschränkung, daß neue Befehle direkt hinter THEN mit einem Doppelpunkt abzutrennen sind.

Falsch ist:

```
IF A = 0 THEN !WALK
14
```

richtig dagegen:

```
IF A = 0 THEN : !WALK
14
```

Bei längeren Programmen oder solchen, die viele Zeichenkettenvariablen verwenden, ist wichtig, als erste Programmzeile diese Anweisung einzubauen:

```
10 POKE 56,140:CLR
```

Damit ist der von Turtle-Grafik reservierte Speicherplatz unantastbar. Jetzt kann man nach Herzenslust eigene Variablen definieren oder Turtle-Befehle benutzen.

Neue Fehlermeldungen

Tritt ein Fehler auf oder ist das Turtle-Basic-Programm zu Ende, schaltet sich automatisch der Textmodus ein.

Hier die neuen Fehlermeldungen (alphabetisch), die ins Programm integriert wurden:

?FORMULA TOO COMPLEX ERROR: Diese schon von Basic bekannte Meldung erscheint bei Turtle-Grafik vor allem bei der !STACK-Funktion; wenn's also z.B. nicht gelingt, die Position des Stackzeigers zu berechnen. Normalerweise liegt ein Systemfehler vor, oder ein anderes Programm verwendet eine von Turtle-Grafik reservierte Speicherzelle.

?OUT OF SCREEN ERROR: Die Turtle hat den gültigen Bildschirmbereich verlassen (x = 0 bis 319, y = 0 bis 199) – oder steht kurz davor.

?STACK OVERFLOW ERROR: Der Stack kann maximal 1637 Werte verkraften. Versucht man, per !PUSH doch noch weitere unterzubringen, erhält man diese Meldung.

?STACK UNDERFLOW ERROR: Im Stapelspeicher ist kein Element mehr vorhanden. Diese Fehlermeldung erscheint beim !PULL-Befehl.

?UNKNOWN ARGUMENT ERROR: Hinter dem Ausrufezeichen steht eine Funktion, die Turtle-Grafik nicht identifizieren kann (evtl. Schreib- oder Eingabefehler).

?UNKNOWN STATEMENT ERROR: unbekanntes Befehlswort hinter dem Ausrufezeichen

Turtle-Grafik (Speicherbelegung)

Adresse	Funktion
$0002	Vorzeichen
$0003 bis $0004	Stackpointer
$0007	Vorzeichen Polarkoordinate
$000c bis $000e	karthesische Koordinate
$000f bis $0010	Polarkoordinate: Radius
$0011 bis $0012	Polarkoordinate: Winkel
$0022 bis $0023	Zeiger in Basic-Programm
$0024	Nr. des Befehls/Speicher für !POLY
$0025	Zähler für !POLY
$0057 bis $0058	Linie: DX
$0057	temporär
$0059 bis $005a	Linie: DY
$005b bis $005c	Linie: XADD
$005d	Linie: YADD
$005e bis $005f	Linie: REST
$0092	Turtle y-Koordinate
$0096 bis $0097	Turtle x-Koordinate
$009b bis $009c	Turtle Winkel
$009e	!RAY-Flag
$009f	Trennzeile !WINDOW
$00a4 bis $00a5	Linie Start x
$00a6 bis $00a7	Linie Ende x
$00a8	Linie Start y
$00a9	Linie Ende y
$00aa	Linienmuster !PATTERN
$00ab	Zeichenmodus !MODE
$00b0	Grafik-Farben !SCREEN
$00b1	Flag: Basic-Fehler
$00b2 bis $00b3	!POLY: Weite
$00b4 bis $00b5	!POLY: Winkel
$00b6	temporär
$00bd	Polar - Karthes.: Vorzeichen X
$00be	Polar - Karthes.: Vorzeichen Y
$00f7 bis $00f8	Polar - Karthes.: X
$00f9 bis $00fa	Polar - Karthes.: Y
$00fb bis $00fe	temporär
$00ff	Speicher für Linie
$0800 bis $8bff	Basic-Bereich
$8c00 bis $8fff	Grafikfarbspeicher
$9000 bis $9fff	unbenutzt
$a000 bis $bfff	Grafik-Bitmap
$c000 bis $c9c7	Programm Turtle-Grafik
$c000 bis $c002	Sprung zum Start
$c003 bis $c005	interner Sprung
$c006 bis $c00d	Zweierpotenzen
$c00e bis $c013	Video-Konstanten
$c014 bis $c045	Grafik-Multiplikationstabelle
$c046 bis $c0a0	Sinustabelle
$c0a1 bis $c10d	Einschaltmeldung
$c10e bis $c18c	Befehlstexte
$c18d bis $c1a5	Funktionstexte
$c1a6 bis $c1d9	Adressen der Befehle
$c1da bis $c1e3	Adressen der Funktionen
$c1e4 bis $c22e	neue Fehlermeldungen
$c22f bis $c9c7	Maschinensprache-Code
$c22f bis $c260	Uplink
$c261 bis $c386	Basic-Interface
$c387 bis $c60f	Turtle-Grafik
$c610 bis $c6b3	Stack-Befehle
$c6b4 bis $c73c	polar – karthes. Koordinaten
$c73d bis $c9c7	Grafiksteuerung
$c9c8 bis $dfff	unbenutzt
$e000 bis $fffd	Stack
$fffe bis $ffff	unbenutzt

Turtle-Demos auf Diskette

Sind Sie neugierig darauf, was Turtle-Grafik kann? Dann sehen Sie sich die Bilder an, die unsere Demoprogramme 1 bis 9 fabrizieren.

Laden und starten Sie zunächst den Turtle-Interpreter, dann die Programmbeispiele (Turtle-Demo 1 bis 9, Abb. 2 bis 10). Abb. 1 zeigt das Demoprogramm »Turtle-Schrift«.

Häufig sind Werte einzugeben, die das Bild günstig beeinflussen. In solchen Fällen genügt es, die vorgegebenen Zahlen einfach mit <RETURN> zu übernehmen.

In Turtle-Grafik ist keine Hardcopy-Routine eingebaut. Jeder Druckerbesitzer hat garantiert ein Utility in der Diskettensammlung, mit dem man Bilder aus dem RAM-Bereich ab $A000 (40960) drucken kann. Falls nicht, läßt sich die Ladeadresse einer gespeicherten Turtle-Grafik per Diskettenmonitor in gewünschte Werte ändern (z.B. $2000). Oder man spürt die Grafik mit einem geeigneten Tool (z.B. Hardmaker im 64'er-Sonderheft 53 oder Hi-Eddi, 64'er-Sonderheft 75) auf, sichert sie und druckt sie aus.

Tips & Tricks

Das Programm kennt auch keine Kreis-Funktion (wie z.B. CIRCLE). Dennoch ist es kinderleicht, Kreise zu erzeugen:

```
10 !INIT
20 FOR I=1 TO 120
30 !WALK 4
```

```
40 !TURNL 4
50 NEXT
60 !FINISH
```

Der Computer muß also 120 x vier Schritte in aktueller Richtung zeichnen, dann vier Grad nach links drehen und weitermachen: voilà– ein Kreis.

Der Vorteil von Turtle-Grafik: alle Grafik-Befehle arbeiten relativ zu den Koordinaten des vorhergehenden Befehls. Verzichtet man auf absolute Positionierungsbefehle (wie !MOVE und !ANGLE), läßt sich das Bild z.B. spiegeln, verschieben, oder um beliebige Winkel drehen.

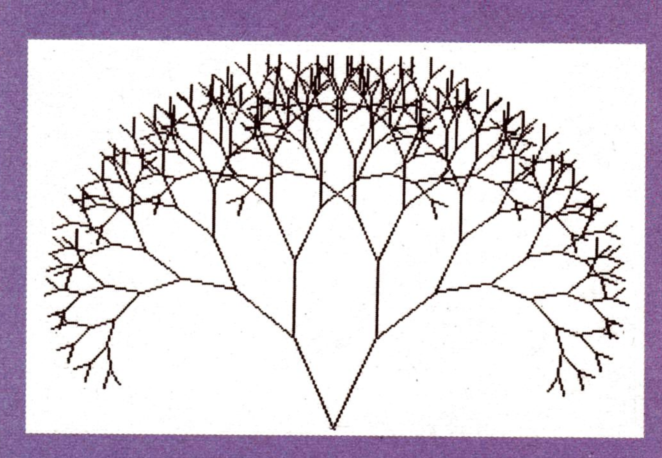
Damit sich z.B. das ganze Bild um sechs Grad im Uhrzeigersinn dreht, setzt man einfach »!TURNR6« Befehl an den Programmanfang – fertig! Da die Schildkröte jetzt bereits von Beginn an auf den Sechs-Grad-Winkel geeicht ist, berücksichtigen alle nachfolgenden Befehle diesen Wert: das Bild erscheint gedreht.

Ähnlich geht's bei der Verschiebung zu. Unser Demoprogramm Nr. 6 (»Tunnel«) ist ein Beispiel dafür. Hier wird zweimal der gleiche komplexe Körper von derselben Routine gezeichnet; allerdings an unterschiedlicher Position und um 180 Grad gedreht (= gespiegelt).

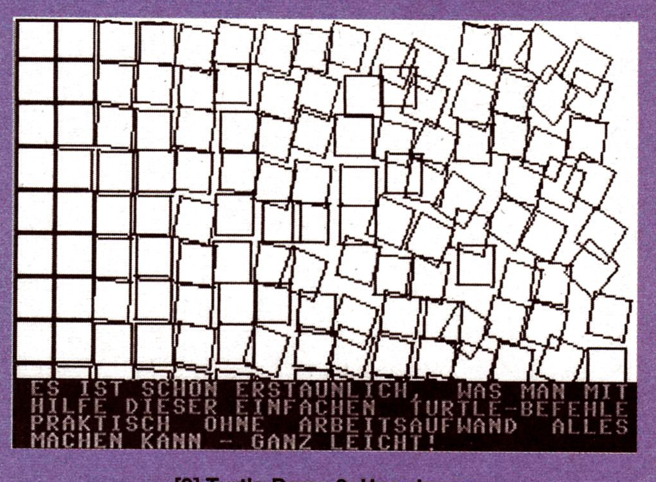
Rekursive Programmierung

Viele Effekte in unseren Demos entstehen durch rekursive Programmierung. Beispiel »Schneeflocke« (Turtle-Demo 5):

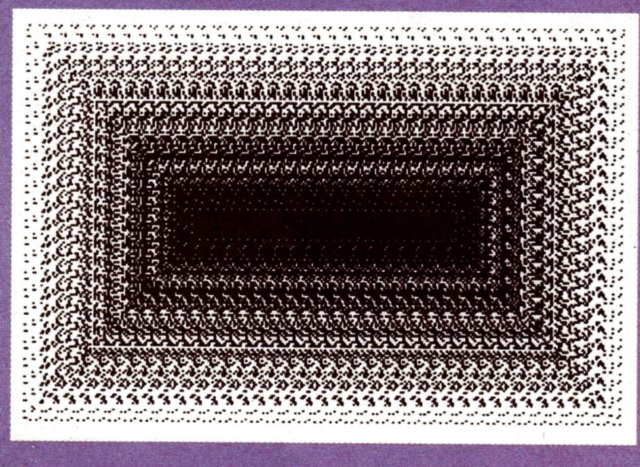
Nehmen wir die Routine zum Zeichnen einer Linie. Sie zieht aber nicht einfach einen Strich, sondern reagiert völlig anders: Erst verläuft die Linie geradeaus (ein Drittel der geplanten Länge). Dann dreht sich die Turtle um 60 Grad (gleichseitiges Dreieck) nach links und zeichnet wieder 1/3 der Länge. Anschließend wendet sie sich 120 Grad nach rechts, zeichnet das nächste Drittel, dreht sich abschließend 60 Grad nach links (jetzt hat die Turtle wieder die ursprüngliche Richtung) und malt des letzte Drittel der vorgegebenen Länge. Wie Sie auf einem Blatt Papier leicht nachprüfen können, hat man dann exakt den Punkt erreicht, zu dem man auch ohne große Umstände gekommen wäre – falls man nur einen Strich gezeichnet hätte. Jetzt ist aber ein Dreieck daraus geworden (besser: ein Muster, das einer Wünschelrute gleicht).



[8] Turtle-Demo 7: Binärbaum



[9] Turtle-Demo 8: Unordnung



[10] Turtle-Demo 9: Patterns

Kurzinfo: Turtle-Grafik

Programmart: Grafikerweiterung
Laden: LOAD "TURTLE-GRAFIK",8
Starten: nach dem Laden RUN eingeben
Besonderheiten: Programm verwendet keine absoluten Koordinaten, sondern Richtungsangaben und Offset-Werte
Benötigte Blocks: 11
Programmautor: Nikolaus M. Heusler

Daran erkennt man: die Routine muß eine Linie viermal zeichnen. Eine Alternative wäre, den !WALK-Befehl einzusetzen. Stattdessen wird diese Routine nochmals aufgerufen, jedoch nur 1/3 der ursprünglichen Länge vorgegeben. Dadurch entstehen immer feinere Muster, die am Ende an eine Schneeflocke erinnern. Dieser Trick berührt schon Gebiete der programmierten Chaos-Grafik (Apfelmännchen, Julia-Mengen usw.).

Bei rekursiver Technik muß man allerdings zwei Dinge beachten:

– Bauen Sie ein Kriterium ein, das die Berechnung abbricht. Irgendwann ist die Länge so geschrumpft, daß nun wirklich gezeichnet werden sollte.

– Vor dem rekursiven Aufruf sollte man stets einen oder mehrere Werte sichern und wieder zurückholen, wenn die Routine fertig ist (also die aktuelle Länge). Dazu läßt sich der Stack verwenden, auf den man per !PUSH, !PULL und !RESUME Zugriff hat.

Wie funktioniert die Stack-Verwaltung? Stellen Sie sich einen Stapel 64'er-Sonderhefte vor. Sie legen nun (!PUSH) das neueste (also Ausgabe Nr. 94) oben drauf. Wenn Sie das Heft wieder runternehmen möchten, müssen Sie den !PULL-Befehl verwenden: er holt sich die oberste Zeitschrift und liest die darauf stehende Zahl (94). Also: Elemente, die zuletzt eingespeichert wurden, werden als erste wieder gelesen – das LIFO-Prinzip (last in, first out).

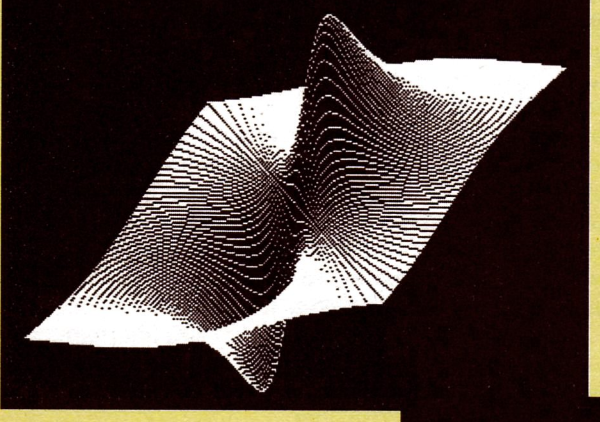
Grau ist alle Theorie – also rein in die Praxis! Wir wünschen ungetrübten Grafikspaß mit unserer Basic-Erweiterung, die ungewöhnliche Bilder mit nur einer Handvoll Befehle erzeugt.

(Nikolaus Heusler/bl)

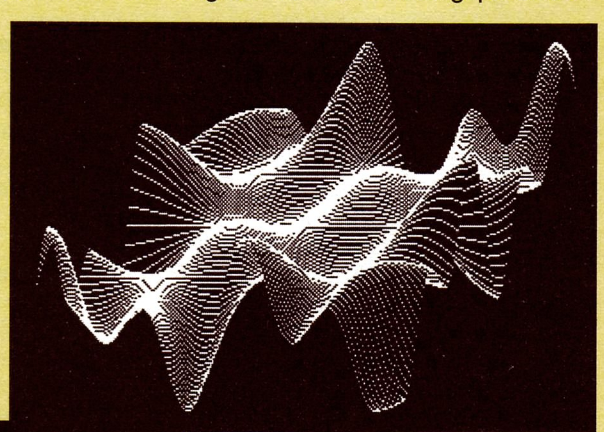
3D-Funktion V1.2

Grafik-Freiräume

Die Formel einer Funktion ist knochentrockene Mathematik – mit unserem Programm aber werden geometrische Gleichungen quicklebendig. Staunen Sie, welch sonderbare Gebilde spezielle Rechenformeln erzeugen!



[1] Grafikdemos zu »3D-Funktion V1.2«: für »bild0« wurde Schrittweite 1 gewählt.



[2] Fantastische Gebilde erfordern lange Rechenzeiten: »bild1«.



[3] »bild2« erinnert entfernt an den Alpen-Hauptkamm ...

Unser Programm erzeugt Intervalle einer reellen Funktion in zwei Veränderlichen auf dem Hires-Schirm des C 64. Auf die so entstandene Grafik muß man anschließend nicht verzichten: Speicher-, Lade- und View-Routinen sind selbstverständlich eingebaut.

Laden Sie das Programm mit:

```
LOAD "3D-FUNKTION
V1.2",8
```

Nach dem Start mit RUN erscheint im oberen Bildschirmbereich ein Menü mit drei Auswahlkriterien, die man durch Eingabe des jeweiligen Buchstabens aktiviert:

<L> Laden: ... lädt nach Angabe des Dateinamens ein Hires-Bild von Diskette (auf unserer Sonderheftdisk finden Sie die Demo-Files »bild0« bis »bild2«) und bringt die Grafik auf den Screen. Achtung: Damit lassen sich auch andere Hires-Bilder laden. Voraussetzung: sie müssen im Hi-Eddi-Format gespeichert sein (32 bis 33 Blocks auf Disk, Ladeadresse bei $2000).

Nach Tipp auf die Leertaste ist man wieder auf dem Menübildschirm.

<B> Berechnen: ... aktiviert die Rechenfunktionen des Programms, welche Grafikkurven auf dem Screen erzeugen. Dazu gibt man die Weltkoordinaten des Intervalls ein (es sind zwei Werte anzugeben: Anfang und Ende, z.B. 1.5 und -1.5), anschließend die Schrittweite in Pixeln (das können Zahlen zwischen 1 und 20 sein). Damit legt man den Abstand der zu berechnenden Punkte auf dem Grafikbildschirm fest.

Um sich ein Bild zur definierten Funktionsformel zu machen, reicht meist schon der Wert »5«. Bedenken Sie: je kleiner der Abstand, desto länger die Rechenzeit! Halbiert man z.B. die Schrittweite, vervierfacht sich die Rechenzeit! Das Programm prüft automatisch, ob die Eingaben zulässig sind (notfalls muß man sie mit korrekten Werten wiederholen!). Anschließend wird der Hires-Modus eingeschaltet und der Graph entsteht auf dem Screen.

Ist die Grafik fertig, drückt man eine beliebige Taste. Man hat nun die Möglichkeit, den Grafikbildschirm nach Angabe eines Dateinamens auf Diskette zu verewigen.

Per <RETURN> geht's ohne Speichern ins Menü zurück.

<E> Ende: ... Programmausstieg ohne Reset. Auf dem Bildschirm erscheint die wichtigste Programmzeile (Nr. 19): sie ist dafür verantwortlich, welche Intervallgrafiken auf dem hochauflösenden Screen erscheinen. Die Voreinstellung des Programms

```
19 Z = COS(50-ABS(X*Y))*45
```

läßt sich jederzeit von Hand ändern. Nach erneutem Programmstart und der Eingabe der Berechnungsparameter dient der Inhalt von Zeile 19 so lange als Rechengrundlage, bis man die Zeile ändert.

Neues Funktionenbeispiel

Holen Sie Zeile 19 mit dem LIST-Befehl auf den Bildschirm. Probieren Sie's mal mit dieser Formel:

```
19 Z=COS(SQR(X*X+Y*Y))*45
```

Im Menüpunkt »Berechnen« trägt man x-Koordinaten von 1,6 bis -1.6 ein, als y-Werte die Zahlen 1,5 und -1,5. Wählt man als Schrittweite »4«, dauert's etwas mehr als drei Minuten, bis die Grafik fertig ist. Entscheidet man sich für »1«, braucht das Bild 16mal länger, also knapp eine Stunde.

In der Basic-Anweisung (Zeile 19) lassen sich selbstverständlich alle Funktionen und Operatoren des Basic 2.0 eintragen (z.B. SIN, TAN, COS, PEEK usw.). Achten Sie aber darauf, daß 3D-Funktion V1.2 keine »Division by zero«-Fehler abfängt und das Programm aussteigt. Das sollte man per IF-THEN-Anweisung (integriert in einer zusätzlichen Programmzeile hinter Nr. 19) ausdrücklich ausschließen!

Viel Vergnügen beim Entwurf toller Grafiken, die lediglich aus einer Funktionsformel entstehen. Unsere Abb. 1 bis 3 zeigen drei Beispiele. (bl)

Kurzinfo: 3D-Funktion V1.2:

Programmart: Grafik-Utility
Laden: LOAD "3D-FUNKTION V1.2",8
Starten: nach dem Laden RUN eingeben
Besonderheiten: Rechenzeiten zwischen vier Minuten und mehr als einer Stunde
Benötigte Blocks: 6
Programmautor: Rüdiger Strey



Das Hauptmenü bietet jede Menge Funktionen, um überdimensionale Screens zu entwerfen

Big Pic V5 – überdimensionale Screens

C 64 als Grafikriese

Wer kennt sie nicht – Spiele wie z.B. »Boulder Dash«, bei denen die Spielfläche größer als der normale Bildschirm des C 64 ist. »Big-Pic V5« ist ein Tool, mit dem man solche Riesengrafiken komfortabel erzeugt.

Maximal 3 x 3 Screens (= 9) kann man lückenlos aneinderhängen, die sich dann per Joystick oder Cursor-Tasten scrollend verschieben lassen. Die Spielelandschaft setzt man aus speziell definiertem oder geändertem Zeichensatz zusammen – in Multicolor oder einfarbig.

Laden Sie das Tool von der Diskette zu diesem Sonderheft:

```
LOAD "BIG-PIC V5",8
```

Gestartet wird mit RUN.

Das Hauptmenü (Abb.) zeigt alle Programmfunktion:

<1>: Zeichensatz normal,
<2>: Zeichensatz bei $2000 (8192),
<3>: Zeichensatz bei $3000 (12288),
<4>: Multicolormodus aus,
<5>: Multicolormodus an,
<6>: Multicolorfarbe 1 aktivieren,
<7>: aktiviert Multicolorfarbe 2,
<G>: normales Zeichen einlesen,
<N>: 2 x 2-Char-Block lesen,
<Z>: Zeichensatz betrachten,
<X>: Programm beenden,
<F1>: Farbe des Bildschirmrahmens,
<F3>: Hintergrund,
<F5>: Schriftfarbe,
<CLR/HOME>: Teilbildschirm (40 x 25 Bytes) löschen.

Merken Sie sich die Infos (oder lassen Sie die entsprechende Seite unseres Sonderhefts neben dem Computer liegen): Die Tastenbelegung erscheint nur unmittelbar nach dem Programmstart und läßt sich erst wieder beim Neustart des Tools aktivieren.

Mit einer beliebigen Taste geht's weiter. Nach dem Hinweis auf die Basic-Anweisung (SYS 51000), die das Programm nach Reset oder Abbruch erneut aktiviert, meldet sich nach ca. drei Sekunden der Editor-Screen. Der ist allerdings vorerst noch mit konfusen Zeichen übersät – per <SHIFT CLR/HOME> macht man reinen Tisch (zumindest für den sichtbaren Bildschirmbereich).

Der Editierbildschirm

In der obersten linken Bildschirmecke blinkt der Editier-Cursor, der sich per Joystick in Port 2 oder mit den Cursor-Tasten über den gesamten Riesen-Screen bewegen läßt. Hinweis: beim horizontalen Scrolling nach links oder rechts sollte sich der Cursor nicht in der obersten oder untersten Bildschirmzeile aufhalten, da das Scrolling auf- bzw. abwärts absolute Priorität besitzt. Der erste Screen verschwindet nun nach links aus dem Sichtfeld. Das Scrollen stoppt erst, wenn Sie den Cursor vom rechten Rand wegziehen oder drei Bildschirme passiert sind. Berührt man den unteren Rand, scrollt das Bild aufwärts: Bewegen Sie jetzt den Editor-Cursor von den Rändern weg, und drücken Sie <Z>: in der oberen Bildschirmhälfte taucht der geänderte bzw. Original-Zeichensatz auf. Steuern Sie nun den Cursor zum gewünschten Zeichen und tippen Sie auf <G>: Das Tool übernimmt den gewählten Charakter, der restliche Zeichensatz erlischt.

Sollten die Zeichenmuster über einen Teil des neu entworfenen Bildes hinausragen, macht das überhaupt nichts: Tipp auf <G> stellt den ursprünglichen Zustand wieder her. Vorsicht: Wenn man das Zeichen noch nicht per <G> übernommen hat, sollte man aufs Bildschirm-Srollen verzichten. Sonst werden nämlich die momentan sichtbaren Zeichensatzmuster automatisch als fester Bestandteil ins Bild eingebaut und unverändert im Gesamtbildschirm übernommen!

Zeichen an beliebiger Stelle plazieren

Per Feuerknopf oder Asterix-Taste <*> läßt sich das gewählte Zeichen beliebig im gesamten Riesen-Screen verteilen – so oft Sie wollen! Um Zeichen zu löschen, übernimmt man mit dem Cursor und <G> ein Leerzeichen.

Zusätzlich ist es möglich, nicht nur ein Zeichen, sondern auch 2 x 2-Char-Blöcke unterzubringen. Stellen Sie dazu auf einem unbesetzten Platz auf dem Screen einen quadratischen, aus vier Zeichen (z.B. Mauersteine) bestehenden Block zusammen. Jetzt bewegt man den Cursor aufs linke obere Zeichen und drückt <N>: der Editor-Cursor verändert seine Gestalt und umfaßt jetzt vier Zeichen! Ansonsten gelten die gleichen Editiermöglichkeiten wie für den normalen Cursor-Block.

Programmhinweise

Verwenden Sie eigene Zeichensätze, sollten die Muster entweder bei $2000 oder ab $3000 im Speicher liegen: diese RAM-Bereiche werden von Big-Pic V5 unterstützt.

Nach dem Laden belegt das Tool den Speicherbereich von $0801 bis $2000 und kopiert sich nach dem Start automatisch nach $C000-$CFFF.

Weitere RAM-Speicher, die das Programm benötigt:
- Zeichensatz: $2000 bzw. $3000,
- Sprites: $0340 bis $03C0,
- Riesen-Screen: von $4000 bis $8000.

Das ergibt ein Bildschirmformat von 120 x 75 Zeichen (der normale Textbildschirm des C 64 hat nur 40 x 25!). (bl)

Kurzinfo: Big-Pic V5

Programmart: Grafik-Tool
Laden: LOAD "BIG-PIC V5",8
Starten: nach dem Laden RUN eingeben
Besonderheiten: Editieren und Verwalten von neun Scroll-Screens. Neustart nach Reset mit SYS 51000.
Benötigte Blocks: 25
Programmautor: Christian Schmelzer

Grafiktester – scannt Lores-Screens

Den Grafikzeichen auf der Spur

Die wenigsten Spiele-Programmierer arbeiten mit kompletten Hires-Screens: Meist verwendet man für die Super-Grafiken geänderte Zeichensätze. »Grafiktester« bringt ans Tageslicht, welche Zeichen geändert wurden.

Was auf den ersten Blick wie ein mühsam Bildpunkt für Bildpunkt entworfenes Multicolorbild aussieht (z.B. mit Koala Painter, Amica Paint usw.), entpuppt sich bei näherem Hinsehen oft als geändertes Zeichenmuster des Original-Commodore-Zeichensatzes. Womit die Leistung solcher Trick-Grafiker nicht geschmälert werden soll: es macht ebenfalls jede Menge Arbeit, den Zeichensatz des C 64 Pixel für Pixel umzustellen – allerdings mit einem gravierenden Unterschied: oft verwendete Zeichen muß man nur einmal ändern und kann sie dann im Bild beliebig oft einsetzen!

Bildschirminhalte scannen

Will man also Teile einer fremden Grafik (z.B. eines Spiels) in eigene Scroll-Screens einbauen, ist es gut zu wissen, aus welchen Standardzeichen die Grafikobjekte zusammengesetzt sind. Um das herauszufinden, gibt's drei Methoden:
– Man lädt den gewünschten Bildschirm mit »,8,1« und schaltet in den Kleinschriftmodus. Alle geSHIFTeten Zeichen erkennt man an den Großbuchstaben (meist sind es aber zu wenig, um weitere Rückschlüsse auf geänderte Zeichen zu erhalten!)
– Man berechnet die gewünschte Adresse des Bildschirmspeichers (z.B. Adresse 1694) und liest sie mit PEEK. Bleibt aber immer noch die Umrechnung in den jeweiligen ASCII-Code.
– ... oder man wählt die komfortabelste Methode: den Grafiktester!

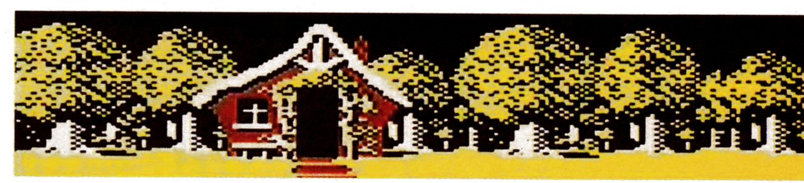
Laden Sie das Utility mit:

```
LOAD "LOADER",8
```

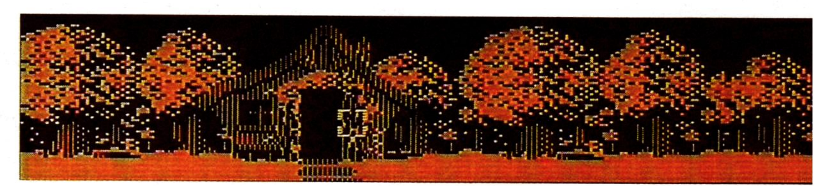
Starten Sie mit RUN. Zunächst möchte das Programm seine Fähigkeiten demonstrieren: Sie werden aufgefordert, zuerst den entsprechenden Zeichensatz (»demofont«), dann den Bildschirminhalt (»demografik«) von unserer Sonderheftdiskette zu laden. Nach kurzer Zeit erscheint der Demo-Screen (Abb. 1, aus dem bekannten C-64-Game »Hexenküche«).

Eingabe- oder Floppyfehler werden abgefangen. Der Zeichensatz wird stets an die Adresse $3000 geladen. Damit »Grafiktester« nicht von überdimensionalen Bildschirmen überschrieben wird, holt man den Grafik-Screen nach Adresse $4000. Nur die ersten 1000 Bytes der Grafik werden ins Bildschirm-RAM kopiert. Zur Weiterverarbeitung stehen an Tastenfunktionen bereit:

<M>: ... umschalten zwischen Multicolor- und Einzelfarb-Modus,
<1>: ... erhöht die Hintergrundfarbe,
<2>: ... schaltet die Multicolorfarbe #1 weiter,



[1] Beispielgrafik eines beliebten Spiels



[2] Die Info-Sprites zeigen exakt, welches Originalzeichen geändert wurde und mit welcher Tastenkombination es erzeugt wird.

<3>: ... Multicolorfarbe #2,
<4>: ... inkrementiert die Vordergrund-Zeichenfarbe ab $D800 (55296).

Achtung: Die Funktionen 2 bis 4 sind nur dann wirksam, wenn man den Multicolormodus (per M) aktiviert hat!
<F1>: ... ruft den Check-Modus auf.

Grafik-Screens durchkämmen

Nach Tipp auf <F1> befindet man sich im Check-Modus. Oben links wartet der Such-Cursor, bis man ihn mit den Cursor-Tasten (nicht per Joystick!) über den Bildschirm jagt. Je nach Code des Zeichens, das unter dem Cursor liegt, sind im oberen Bildschirmbereich drei Info-Sprites zu erkennen:
– ganz links: Kästchen mit inversem R (das gecheckte Bildschirm-Byte ist ein reverses Grafikzeichen!)
– Mitte: Kästchen mit den Buchstaben SH bzw. Commodore-Symbol (es ist ein SHIFT-Zeichen oder eines, das sonst mit der Commodore-Taste erzeugt wird),
– rechts: das eigentliche Tastaturzeichen.

Beispiele: Bewegen Sie den Check-Cursor auf die linke Seite der Dachspitze des Hexenhauses. Sofort erscheint der SH-Rahmen, daneben der Schrägstrich links: das ist die Tastenkombination, die die Spiele-Programmierer in dieses Grafikelement umgewandelt haben (Abb. 2). Oder: stößt man aufs Ausrufezeichen <!>, bringt der Bildschirm: SH 1. Wenn Sie mit dem Cursor über die unterste Bildschirmzeile fahren (Grünfläche), fällt Ihnen schnell auf, daß die Wiese überwiegend aus dem großen O besteht. Screen-Teile, die aus Null-Bytes oder Leerzeichen bestehen (Code 32), bringen verständlicherweise nur leere Sprite-Rahmen.

Die Anzeige durch Info-Sprites wurde absichtlich gewählt, um nicht einen Teil des Screens per Rasterinterrupt abtrennen zu müssen und so die Grafik unnötig zu verkürzen. Wenn die Sprites im linken Bildschirmteil stören, verbannt man sie kurzerhand per <F5> nach rechts. Mit <F7> ist es wieder so wie vorher. Per <F1> kann man den Check-Modus verlassen und das Lademenü aufrufen – zurück in den Such-Bildschirm geht's aber erst wieder nach dem Laden einer Font- und Grafikdatei. Selbstverständlich kann das Utility bei echten Hires-Grafiken nicht eingesetzt werden. (bl)

Kurzinfo: Grafiktester

Programmart: Utility
Laden: LOAD "LOADER",8 oder
LOAD "GRAFIKTESTER",8,1
SYS 2088
Starten: mit RUN oder SYS 2088
Besonderheiten: arbeitet im Single- und Multicolormodus
Benötigte Blocks: 23
Programmautor: Reiner Riemer

Kurz und bündig

Wer sagt, daß man unbedingt umfangreiche Tools braucht, um tolle Spiele zu kreieren? Unsere vier Hilfsprogramme verbrauchen kaum Platz auf der Disk und machen's ebenso gut!

Das Game-Maker-Tool besteht aus vier Utilities, die erstaunlicherweise zusammen nicht mehr als zehn Blocks auf Diskette belegen. Es sind:
- 1. GAME,
- 2. RAM EDITOR,
- 3. COL.+SPR.ED,
- 4. CHAR.EDITOR.

Mit dem zuletzt genannten Programm geht's los: Voraussetzung für ein Spiel mit fetziger Grafik ist zunächst ein toller Zeichensatz, aus dem die Spielelandschaft besteht.

Der Zeichensatz-Editor

Laden Sie das Utility mit:

```
LOAD "4. CHAR.EDITOR",8
```

Nach dem Start mit RUN wird der Originalzeichensatz ab $D000 automatisch nach $2000 kopiert. Der Cursor wartet jetzt auf die Eingabe des gewünschten Zeichens, das Sie verändern möchten. Erlaubt sind alle 256 Charakters, die der Bildschirmspeicher anzeigen kann. Nach <RETURN> bringt der Screen das 8 x 8 Zeichen große Editorfeld (Abb. 1). Mit jeder beliebigen Zahlen- oder Buchstabentaste (inkl. <SHIFT SPACE>) setzt man ein Pixel, per <SPACE> wird's gelöscht. Will man ein Zeichen entwerfen, das später im Multicolormodus erscheinen soll, sind die entsprechenden Bitcodierungen zu beachten!

Drückt man in der untersten Pixelzeile <RETURN>, wird das Zeichen verändert und die entsprechenden acht Byte-Werte erscheinen auf dem Bildschirm.

Der Sprite-Editor

Das nächste Werkzeug entwirft die fürs Spiel notwendigen Sprite-Muster:

```
LOAD "3. COL.+SPR.ED",8
```

Startet man mit RUN, wird der Speicher ab $1000 (4096) zur Aufnahme des Sprite-Musters vorbereitet (nach dem späteren Start des fertigen Spiels werden die Sprite-Pixel nach $02C0 (704) verschoben).

Bei der folgenden Frage hat man die Wahl: »Sprite-Editor oder Zeichenfarben setzen?«. Nach Tipp auf <S> muß man sich entscheiden, ob man Multicolorsprites entwerfen will. Dann sind die Code-Zahlen der jeweiligen Sprite-Farben anzugeben. Anschließend taucht das Sprite-Editorfenster auf.

Per Taste <Z> vergibt man die gewünschte Farbe (Codes 0 bis 15) ans gewählte Zeichen. Die behält der Charakter während des gesamten Spielverlaufs.

Der RAM-Editor

... bereitet den Speicher auf 30 Spielfeld-Bildschirme vor:

```
LOAD "2. RAM EDITOR",8
```

Den »Syntax-Error«, der sich nach dem Start mit RUN meldet, dürfen Sie getrost vergessen: die Programmfunktionen werden davon keineswegs beeinträchtigt! Bedeutend wichtiger ist die Basic-Zeile, die man anschließend im Direktmodus eingeben muß:

```
FOR A=10240 TO 40800: POKE A,32: NEXT
```

Es dauert etwa 2 Minuten, bis sich der Cursor wieder mit READY meldet – aber das Bildschirm-RAM für die maximal 30 Game-Screens ist jetzt optimal vorbereitet! Mit den Tasten <F1> bzw. <F3> kann man im RAM spazierenfahren (Abb. 2). Der Screen scrollt nach oben oder unten (nicht seitlich!). Tragen Sie nun Ihre Level-Elemente mit den Tasten der geänderten Zeichen ein oder löschen Sie solche mit <SPACE>, die Sie nicht brauchen. Der gesamte aktuelle Screen-Ausschnitt wird per <SHIFT CLR/HOME> vom Byte-Müll befreit. Wurde der sichtbare Bildschirmteil verändert, muß man ihn per <F5> in den Speicher übernehmen.

Sind Sie mit Ihrem Werk zufrieden (nachdem also der gesamte Hyper-Screen geändert wurde), drücken Sie <F7>: vier Fragezeichen erscheinen als Aufforderung, den gewünschten Dateinamen einzugeben. Nach <RETURN> werden alle bislang beschriebenen Manipulationen (Zeichensatz, Sprites und Screens) als Gesamtdatei auf Disk ausgelagert (Achtung: das File verbraucht mehr als 150 Blöcke auf Diskette!). Rechnen Sie nach: 30 Screens pro 1000 Bytes (ohne Farben!) sind schon 30000 Byte ...

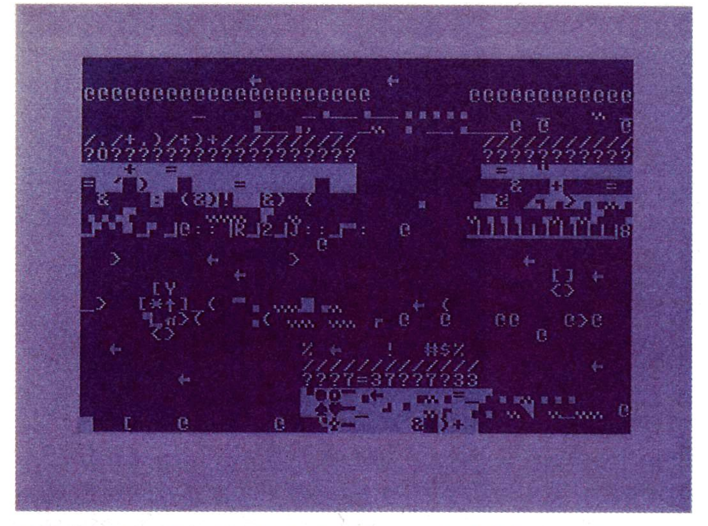
Das Spiel ist fertig!

Damit man sich von der Qualität seines Kunstwerks überzeugen kann, lädt man das eigentliche Steuerprogramm des Game-Maker-Tools:

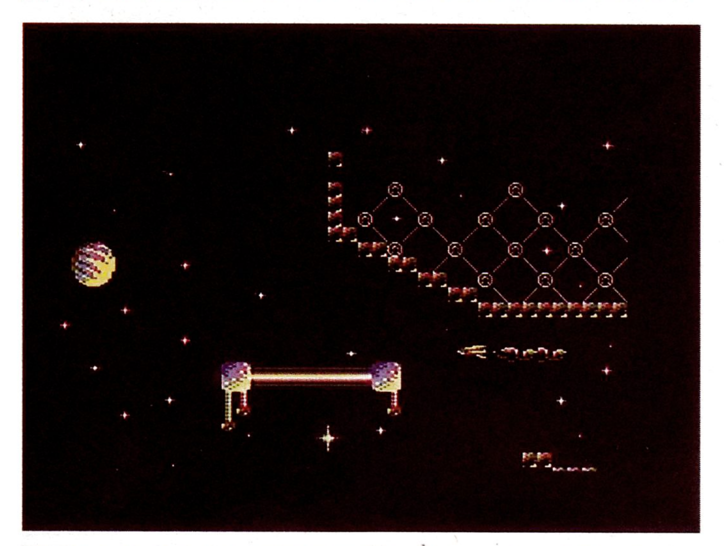
```
LOAD "1. GAME",8
```



[1] Zum Editieren neuer Zeichensätze eine Taste drücken



[2] So sieht die Levellandschaft mit normalen Zeichenmustern aus



[3] Game-Demo1: kniffliges Labyrinth auf einem fernen Planeten

Per Feuerknopf (Joystick Port 2) geht's los: oben befindet sich das Sprite, das Sie zuvor entworfen haben. Die Bildschirme mit der Spielelandschaft (geänderte Zeichensätze per Utility »4. Char.Editor«) scrollen unaufhörlich nach oben. Steuern Sie das Sprite behutsam durch die Landschaft: berührt es eines der geänderten Zeichen, ist das Spiel vorbei – die Meldung »Game over« erscheint. Dieser Text ist in zwei Zeilen aufgeteilt: Die Bytes der ersten Zeile stehen im Bereich von $9F60 (40800) bis $9F87 (40839), die Farbwerte in $9F88 bis $9FAF (40840 bis 40879). Der Text der zweiten Meldung (»Press Button«) ist im Bereich von $9FB0 (40880) bis $9FD7 (40919) abgelegt, dahinter die dazugehörigen Farben ($9FD8 bis $9FFF). Achtung: Der Text wird als Bildschirmcode interpretiert (nicht als ASCII!).

Falls Sie mit dem vorgegebenen Beispieltext nicht einverstanden sind, läßt sich dieser jederzeit ändern:

```
10 INPUT CHR$(147); A$: REM TEXT EINGEBEN
20 REM UND DIE FARBEN NICHT VERGESSEN!
30 FOR A=0 TO 39
50 POKE 40800+A,PEEK(1024+A): REM TEXT
ZEILE 1
60 POKE 40840+A,PEEK(55296+A): REM FARBEN
80 POKE 40880+A,PEEK(1064+A): REM TEXT
ZEILE 2
90 POKE 40920+A,PEEK(55336+A): REM FARBEN
100 NEXT A
```

Das funktioniert selbstverständlich nur, wenn man zuvor das per <F7> gespeicherte Spiel (s. »2. RAM EDITOR«) ins Computer-RAM geholt hat.

Dazu ein Beispiel für die Ladeanweisung:

```
10 IF A=0 THEN A=1: LOAD "(Spielname)",8,1
20 REM LAEDT DIE PER RAM-EDITOR GESICHERTEN
30 REM DATEN DES GAMES
40 LOAD "1. GAME",8
50 REM STEUERPROGRAMM LADEN UND AKTIVIEREN
```

Das Utility enthält diverse Adressen, in denen die Farben abgelegt sind:

- $0ACB (2763): Hintergrundfarbe,
- $0AD0 (2768): Sprite-Farbe,
- $0ADA (2778): Multicolorfarbe 1
- $0AD5 (2773): Multicolorfarbe 2
- $0ADF (2783): Multicolor-Sprite
- $0A37 (2615): Rahmenfarbe (0 = aus, 1 = an).

Die Werte sind leicht zu ändern: Laden Sie »1. GAME« wie gewohnt und verankern Sie die gewünschten Zahlen per POKE-Anweisung in den genannten Speicherstellen. Anschließend sollte man das Utility wieder auf Diskette sichern (SAVE "1. GAME",8).

Programmhinweise

Selbstverständlich ist es ungeheuer arbeitsaufwendig, Spielelandschaften (geänderte Zeichen) und Sprites in einem Rutsch fürs geplante Spiel zu entwerfen. Das kann man auch in mehreren Arbeitsgängen erledigen. Wichtig ist aber stets, nach getaner Arbeit für den jeweiligen Teilabschnitt das Utility »2. RAM EDITOR" zu laden und den entsprechenden Bereich per <F7> zu sichern. Den holt man vor der nächsten Computersitzung wieder in den Speicher, z.B.:

```
LOAD "TEILGAME",8,1
```

Nach der Eingabe von NEW laden Sie die benötigten Utilities (Char-, Sprite- und RAM-Editor), scrollen per <F1> und <F3> durch die Screens und machen dort weiter, wo Sie das letztemal aufgehört haben.

Auf der Diskette zu diesem Sonderheft finden Sie zwei Demo-Programme, die mit dem Utility-Quartett erzeugt wurden. Sie lassen sich wie normale Basic-Programme laden und mit RUN starten.

GAME-DEMO1 wird zunächst entpackt, dann erscheinen die beschriebenen Meldungen auf dem Bildschirm. Stecken Sie den Joystick in Port 2, drücken Sie den Feuerknopf – und schon kann's losgehen (Abb. 3): Steuern Sie das Raumschiff geschickt durchs Labyrinth eines futuristischen Planeten und vermeiden Sie jegliche Berührung mit den geänderten Zeichen des Bildschirmvordergrundes. Das Spiel läßt sich zwar per <RUN/STOP RESTORE> abbrechen, anschließend sollte man aber einen Reset ausführen. Wer keinen Resetknopf besitzt, muß den Computer aus- und wieder einschalten.

GAME-DEMO2 (»Bubble«) benutzt lediglich einen Bildschirm. Die Scroll-Bewegung wird durch die blauen Sprites simuliert, die es auf den gelben Ball am oberen Bildschirmrand abgesehen haben. Auch den muß man gekonnt an den Angreifern vorbeimanövrieren.

Obwohl die Game-Maker-Utilities spartanisch kurz sind, bieten Sie dennoch alle Voraussetzungen, um umfangreiche Levelfelder zu entwerfen. Der Riesenbildschirm besteht aus 30 Screens, die nahtlos ineinander übergehen und von unten nach oben scrollen. Das Steuerprogramm »1. GAME« enthält die entsprechenden Scroll- und Joystick-Routinen.

Viel Spaß beim Entwurf eigener Spielelandschaften und Labyrinthe! (bl)

Kurzinfo: Game-Maker

Programmart: 4 Utilities, um Spiele zusammenzustellen
Laden: LOAD "4. CHAR.EDITOR",8
Starten: nach dem Laden RUN eingeben
Besonderheiten: jedes Programm läßt sich separat laden und starten
Benötigte Blocks: 10 (alle Files)
Programmautor: Sven Forstmann

C-64-Amiga-Connection

TransFile

Das Tool ermöglicht Datenübertragung zu jedem Computer, der mit einer Standard-Parallel-Schnittstelle ausgerüstet ist. In dieser Beschreibung wird als Gegenstelle ein Commodore Amiga benutzt.

Die Systemanforderungen für den Transfer von Files vom C 64 zum Amiga und umgekehrt sind dabei minimal:
- C 64 mit Floppy
- Commodore Amiga mit Workbench (Version 1.X, 2.X)
- Parallelkabel zum Verbinden des C 64 mit anderem Computer (s. Tabelle)
- TransFile für C 64
- ParRead für Amiga zum Empfang von Daten (s. unten)

Vor dem Start muß das Kabel (s. Tabelle) in ausgeschaltetem Zustand der Rechner angebracht werden. Der Empfangscomputer wird immer als erstes in Bereitschaft gesetzt. Probleme mit Parallel-Floppy-Speedern sollten nicht auftreten. Das System arbeitet mit Dolphin-DOS und Kickstart 2.0, wenn der Amiga während des Zugriffs des C 64 auf Disk nicht empfangsbereit gesetzt wird. Bei Kickstart 1.X sollte auf den Speeder verzichtet werden. Nach dem Start wird zuerst der Textfilter »f.bootfilter« nachgeladen. Diese Datei muß sich auf der Diskette befinden. Sollte kein Filter vorhanden sein, kann man das Programm mit »sys 2067« starten. Benennt man den meisten benutzten Filter »f.bootfilter«, so wird dieser automatisch nachgeladen.

Die Menüs

F1 – File senden: Bevor dieser Punkt angewählt wird, muß man ein File in den C 64 laden. Der Amiga wird in einer SHELL mit dem Befehl »ParRead <file>« (s. Kasten) empfangsbereit gesetzt. Erst danach darf die Übertragung angewählt werden. Der Amiga speichert dann die Daten in einer Datei mit Namen <file> ab. Sollte der Zeichenfilter aktiviert sein, berechnet TransFile erst einmal, wie lang das gefilterte File wirklich ist, dann wird es gesendet.

F3 – File empfangen: Hier lassen sich Daten vom Amiga zum C 64 übertragen. Zuerst muß am C 64 der Empfang angewählt werden. Dann gibt man in einer Amiga-SHELL den Befehl »cp <file>« ein.

F5 – Disk Menue: Hier werden alle Ein- und Ausgaben auf der C-64-Seite realisiert.

F1 laden vollständig: Ein C-64-File wird in den Speicher geladen und dabei die Ladeadresse mit eingelesen. Lädt man also ein Basic-Programm, werden die Bytes $01 und $08 für die Ladeadresse $0801 mit geladen. So kann man z.B. eine Sicherheitskopie von einteiligen Files erzeugen.

F3 laden ohne Ladeadresse: Die ersten zwei Bytes werden nicht gelesen. Hat man z.B. einen Text, welcher normal nach $0800 geladen wird, wählt man diese Funktion, da diese Bytes für die reine Text-Information unwichtig sind.

F5 speichern mit extra Ladeadresse: Haben Sie einen Text empfangen, der als File mit Ladeadresse gespeichert werden soll, muß diese Funktion gewählt werden. Die Ladeadresse ist automatisch $0801.

F7 speichern wie empfangen: Die Daten werden direkt ohne Ladeadresse gesichert. Will man also ein Programm zurückübertragen (s. F1 – laden vollständig) oder ein IFF-Bild sichern, ist diese Speicherfunktion nötig, damit das File nicht verfälscht wird.

F2 Filter laden: Ein früher abgespeicherter Zeichen-Filter kann wieder geladen werden. Es werden auch Filter von anderen Programmen akzeptiert. Einzige Bedingung: Der Filter benötigt eine Ladeadresse wie jedes andere Programm auch und die darauf folgenden 256 Byte entsprechen den C-64-ASCII-Codes 0 bis 255. Ein im Speicher befindlicher Filter wird dabei natürlich gelöscht!

F4 Filter speichern: Der im Speicher befindliche Text-Filter wird gespeichert. Es empfiehlt sich, daß vor den Namen ein »f.« gestellt wird, so daß man auch nach längerer Zeit noch weiß, daß dieses File ein Filter war. Die Blocklänge auf Disk beträgt dabei zwischen zwei und neun Blöcken. Hat man im Filter also nur die erste Spalte benutzt, wird nur die erste Spalte gespeichert, bei drei Spalten die ersten drei usw. Es empfiehlt sich also, nicht benutzte Spalten zu löschen, so daß nur noch $00 in den Spalten steht.

D Directory: Der Inhalt der Diskette wird angezeigt.

F2 – Zeichen-Filter editieren: In diesem Editor kann man festlegen, in welche Bytes der entsprechende C-64-ASCII-Code übertragen wird. Dabei können bis zu acht Zielbytes angegeben werden. Z.B. kann man für das Steuerzeichen Delete ($14) die Zeichen »{DEL}« angeben, so daß man auf dem Amiga dann die nicht darstellbaren Codes als Klartext erhält.

Es können nur Hexadezimalzahlen eingegeben werden. Der Cursor wird mit den Cursor-Tasten bewegt. Mit der RETURN-Taste gelangt man an den Anfang der nächsten Zeile. <F1> kopiert die Zeile in einen Puffer, welchen man mit <F3> wieder ausgeben kann. Mit <F5> und <F7> lassen sich die Bytes einer Zeile rotieren, so daß man Zeichen einfügen kann. Gibt man den Code $00 ein, wird dieses Zeichen nicht übertragen. Der Code $00 läßt sich jedoch durch einen anderen ersetzen. So kann man unerwünschte Codes einfach wegfallen lassen. Mit <SHIFT-C> löscht man eine ganze Spalte. <SHIFT-I> initialisiert die erste Spalte mit den darstellbaren Zeichen, die Steuercodes werden alle auf $00 gesetzt. Mit <RUN/STOP> verläßt man den Editor. Als Beispiel

Das Amiga-Empfangs-Programm »ParRead«

Dies ist ein kleines C-Programm (s. Kasten mit Listing), welches den Daten-Empfang für den Amiga ermöglicht. Die Bedienung ist sehr einfach. Man muß das Programm mit einem C-Compiler übersetzen und kopiert dann den erzeugten Code mit dem Befehl

```
copy ParRead to c:
```

in das Befehlsverzeichnis der Systemdiskette oder Festplatte. Danach steht der Befehl über die SHELL zur Verfügung (s. F1 – File senden). Es werden bei Fehlbedienung ausführliche Fehlermeldungen ausgegeben. Das Programm läßt sich für alle möglichen Übertragungen anwenden, also auch von PC zu Amiga usw., das richtige Kabel vorausgesetzt.

Dabei muß in den ersten vier gesendeten Bytes die Länge des Files angegeben werden:

Länge = 1*Byte0+256*Byte1+65536*Byte2 (+16777216*Byte3 nicht implementiert)

Dies sollte für alle Anwendungen ausreichen. Danach werden die Daten-Bytes des Files übertragen. Das Programm läuft im Multitasking, so daß außer Drucken andere Programme laufen können. Das komplette Programm finden Sie auf der Programm-Service-Diskette des Amiga-Magazins Ausgabe 11/93.

Der Befehl »cp« steht als Ersatz für die Befehlsfolge »copy <file> to par:«. Man fügt bei Workbench 1.3 in »s:shell-startup« und bei Workbench 2.x, 3.x in »s:user-startup« folgenden Befehl ein:

```
alias cp copy [] to par:
```

So spart man sich etliche Eingaben, wenn man öfters ein File überträgt. Verändert man ungern etwas an seinen Startup-Files, kann man auch direkt den Copy-Befehl eingeben.

Tabelle: Die Kabel-Belegung

Amiga	C 64
1-Strobe	B-Flag
2-D0	C-PB0
3-D1	D-PB1
4-D2	E-PB2
5-D3	F-PB3
6-D4	H-PB4
7-D5	J-PB5
8-D6	K-PB6
9-D7	L-PB7
10-ACK	8-PC2
22-GND	A-GND

Außerdem muß am Amiga-Kabel der Pin 11 (Busy) und 12(Pout) mit Masse Pin 18 verbunden werden, damit der Konverter auch mit OS2.0 und höher funktioniert.

soll der beiliegende Filter »f.startool« für Startool-Programmtexte dienen: Der Aufbau des Quelltextes besteht aus den normalen Zeichen für Labels und aus Tokens für die Befehle. Für den Assembler-Befehl »BRK « wird das Token $80 erzeugt. Der Amiga kann mit $80 jedoch nichts anfangen, da dies ein nichtdarstellbarer Code ist. Also gibt man für $80 die Bytes »09 42 52 4b 20 00 00 00« ein. Wird nun der Code $80 übertragen, erzeugt der Filter die Zeichenfolge $09 $42 $52 $4b $20. Der empfangene Text erhält an dieser Stelle dann einen Tab-Befehl ($09) zum Einrücken und die Zeichen »BRK«. Der Filter erzeugt für jeden Code einen Klartext, so daß man die Texte mit jedem Editor bearbeiten kann. Dies läßt sich im Prinzip auf jede Art von Text anwenden, außer auf Basic-Programme. Diese enthalten nämlich Zeilennummern im Lobyte-Hibyte-Format. Man muß Basic-Texte erst auf Disk leiten:

`OPEN 1,8,1,»file«: CMD 1: LIST CLOSE 1`

Dann kann man aber einen Filter verwenden, der für die Steuerzeichen Klartexte erzeugt. Ein Nachteil sollte jedoch beim Startool-Filter nicht unerwähnt bleiben: Hat man in Strings direkte Steuercodes angegeben, so werden diese leider als Tokens interpretiert. Dies läßt sich umgehen, indem man für die Steuercodes die Byte-Werte angibt. Beim Empfang von Daten wird nur die erste Spalte des Filters verwendet. Befindet sich das Byte nicht in der Tabelle, wird es ausgelassen. Ein umgewandelter Startool-Text läßt sich aufgrund mangelnder Eindeutigkeit der Daten nicht mehr zurückwandeln, aber eine komplexe Wandlung vom Amiga zum C 64 werden die meisten User auch nicht wollen.

F4 – Sende-Filter an/aus: Diese Option muß bei Bildern, Programmen und Backup-Files ausgeschaltet sein.

F6 – Empfangs-Filter an/aus: Siehe »F4«

F8 – Master-Text modifizieren: Das Master-Text-Format hat einen kleinen Fehler: Nach einem Absatzzeichen ($8e) wird der Text mit Space ($20) bis zu vollen 80 Zeichen aufgefüllt. Wendet man nun direkt einen Filter an, so hat man lauter Space-Zeichen nach einem Absatz vor dem Text. Mit dieser Funktion kann man die Space-Zeichen durch Nullbytes ersetzen lassen. (Peter Steinseifer/lb)

Das Empfangs-Programm »ParRead« für den Amiga muß mit einem Compiler in ein lauffähiges Programm übersetzt werden

```
#include <libraries/dos.h>

#undef ANSI  /*wenn Ansi-C, dann define ANSI*/

ULONG DosBase,datei,datei2;
UBYTE *zeichen;
long int groesse;

#ifdef ANSI
void fail(char * why)
#else
void fail (why)
char *why;
#endif
{
        printf("Fehler: %s\n",why);
        if (datei!=NULL) Close(datei);
        if (DosBase!=NULL) CloseLibrary(DosBase);
        if (zeichen!=NULL) FreeMem(zeichen,groesse);
        exit (FALSE);
}

#ifdef ANSI
void main (int args, char **filename)
#else
void main (args,filename)
int args;
char ** filename;
#endif
{
        unsigned char first[3];

        printf("\nParRead c1993 Peter Steinseifer\n");

        DosBase=OpenLibrary("dos.library",0);
        if (DosBase==NULL)
           fail("dos.library konnte nicht geöffnet werden");

        datei=Open("PAR:",MODE_OLDFILE);
        if (datei==NULL)
           fail("parallel.device konnte nicht geöffnet werden");

        if (args<2)
           fail("kein file angegeben");

        if (args>2)
           fail("zu viele parameter");

        printf("lese Daten vom Parallel-Port...\n");

        Read(datei,&first[0],4);

        groesse=first[0]+256*first[1]+65536*first[2];

        printf("File-Länge:%d\n",groesse);

        zeichen=(char *)AllocMem(groesse,0);
        if (zeichen==NULL)
           fail("kein freier Speicher");

        Read(datei,&zeichen[0],groesse);

        printf("schreibe Daten nach %s\n",filename[1]);

        datei2=Open(&filename[1][0],MODE_NEWFILE);
        if (datei2==NULL)
           fail("File kann nicht geöffnet werden");

        Write(datei2,&zeichen[0],groesse);

        Close(datei2);
        FreeMem(zeichen,groesse);
        Close(datei);
        CloseLibrary(DosBase);

        printf("Datenübertragung beendet.\n\n");

}
```

Tips & Tricks

Mit dem Filter »f.c64ascii« werden die Bildschirm-Codes in Klartext umgewandelt. Zu Gunsten der Übersichtlichkeit, sollte man einmal folgende Codes für den Doppelpunkt ($3a) eingeben: 0a 20 20 20 3a 00 00 00. Steht mehr als ein Befehl in einer Zeile, wird der nächste Befehl einfach in einer neuen Zeile mit drei Leerzeichen vorweg ausgegeben. So erhält man aus undurchsichtigem Spaghetti-Code ein sauber geordnetes Listing. Tritt allerdings der Doppelpunkt innerhalb von Anführungszeichen auf, wird der String ebenfalls getrennt. Doch dies kann man ja mit einem Editor beseitigen, was meist weniger Arbeit bedeutet, als das Listing von Hand zu zerstückeln.

Man kann mit dem Filter auch seine Texte verschlüsseln. Dazu erstellt man sich einen Filter, der nur eine Spalte benutzt. Danach vertauscht man einfach ein paar Zeichen, z.B. das »e« mit Space, usw. Die Verteilung muß jedoch eindeutig sein. Man darf also jedes Zeichen nur einmal verwenden. Schickt man den Text an den anderen Computer, wird er ziemlich verstümmelt. Kopiert man den Text wieder zurück, sollte er wieder entschlüsselt sein. Ohne diesen Filter bekommt man dann nur unleserliches Zeug zurück. Man sollte das aber auf jeden Fall erst mal ausführlich testen! Das Nullbyte $00 darf dabei natürlich keine Verwendung finden (deshalb kann man auch keine Programme verschlüsseln).

Die File-Länge sollte 223 Blöcke nicht überschreiten, da der Rest nicht mehr geladen werden kann. Dies sollte aber für die normalen Anwendungen ausreichen.

Wenn der Bootfilter ersetzt wird, ist darauf zu achten, daß der Filter noch unter einem anderen Namen auf der Disk ist, denn wenn man versehentlich einen wichtigen Filter löscht, kann das schon einiges an Arbeit bedeuten.

Kurzinfo: TransFile

Programmart: File-Konverter für C 64 zu Amiga/PC
Laden: LOAD"TRANSFILE",8
Starten: RUN
Besonderheiten: Kopiert Daten vom C 64 zum Amiga/PC bzw. umgekehrt über Parallelkabel
Benötigte Blocks: 36
Programmautor: Peter Steinseifer

... wie in der ersten Reihe !

Bewegte Grafik, die wie ein Spielfilm auf dem Monitor abläuft: der C 64 ist geradezu prädestiniert dafür! Vorausgesetzt, die Software paßt ...

Es muß nicht immer Hires-Grafik sein. Unser Tool faßt 2 x 2 Bildpunkte quasi zu einem Riesenpixel zusammen (vergleichbar mit den Blockgrafikzeichen <CBM D>, <CBM C> usw.). Durch geschickte Verteilung aller 16 möglichen Farben entstehen damit Super-Grafiken mit einer Screen-Auflösung von 64 x 32 Makro-Pixel.

Maximal 48 verschiedene Bilder lassen sich im RAM-Bereich $A000 bis $FFFF unterm Basic-, bzw. Kernel-ROM plazieren, die in Sekundenbruchteilen eingeblendet werden und damit umfangreiche Animationssequenzen zur Verfügung stellen. Sie können bis zu 127 Schritte umfassen (oder lassen sich in mehrere bis zu dieser Gesamtschrittlänge aufteilen). Für jede Position läßt sich eine individuelle Verzögerung einstellen. Pro Screen-Seite (Page) benötigt das Programm lediglich 512 Byte (inkl. Farbwert).

[1] Zeichenblatt auf dem Bildschirm: Animationsbilder per Joystick entwerfen

Allerdings bleibt es einem nicht erspart, die zur Animation notwendigen Grafiken zunächst einmal zu entwerfen. Laden Sie dazu das erste Tool von der Diskette:

```
LOAD "EDITOR 2X2",8
```

Nach dem Start mit RUN erscheint der Editor-Screen (Abb. 1). Sämtliche Funktionen sind auf bestimmte Tasten gelegt (s. Tabelle). Zeichnen Sie per Joystick in Port 2 die gewünschten Bilder. Mit den Cursor-Tasten schaltet man zum nächsten Screen. <RUN/STOP> dient als Endekennzeichen für die vorgesehenen Animationsgrafiken. Um sie zu speichern, drückt <S>. Das Präfix »S.« wird automatisch vor den Dateinamen der Animation gesetzt.

Um sich mit den Programmfunktionen des Editors vertraut zu machen, sollten Sie die Sequenz »S.Example« von der Diskette zu diesem Sonderheft laden und die Editiertasten einzeln ausprobieren.

Film ab !

Damit vor allem Basic-Programmierer etwas von unserem Movie-Show-Generator haben, wurde eine spezielle Basic-Erweiterung entwickelt, die allerdings als Maschinenprogramm zu laden ist (am besten in der ersten Programmzeile des eigenen Basic-Listings):

```
LOAD "PLAYER 2X2",8,1
```

Gestartet wird mit SYS 39680.

Das Basic 2.0 des C 64 wurde um neue Anweisungen erweitert, die stets mit dem Ausrufezeichen beginnen:

!lade nm$: ... holt die Animationssequenz, deren Name in der Variablen nm$ gespeichert wurde, von Disk in den RAM-Speicher ab $A000 (40960). Achtung: es ist nicht möglich, den Dateinamen im Klartext (z.B. »s.example«) anzugeben!

!play x,y: ... läßt den Film ab Position x bis y ablaufen,

!show x: ...zeigt die Animationsgrafik x

!draw p, x, y, c: ... setzt in Page p einen Grafikpunkt (= 2 x 2 Pixel!) an den Koordinaten x,y in Farbe c. Ist c größer als 15, aber kleiner als 32, wird der Punkt gelöscht.

!fill p,c: ... löscht Grafikseite p und füllt sie mit der Farbe c.

!xxyy x,y: ... definiert die aktuellen Koordinatenwerte xx/yy neu und verlegt sie nach x,y.

Als praktisches Beispiel dient unser Demo-Programm:

```
LOAD "FAST LITTLE DEMO",8
```

Nach dem Start mit RUN lädt der Computer die Basic-Erweiterung und initialisiert sie (sonst würden die neuen Basic-Befehle nur Fehlermeldungen erzeugen!). Lehnen Sie sich bequem zurück und betrachten Sie die Movie-Show, die aus 32 Bildern besteht. Das Studium des Programmlistings gibt Ihnen wertvolle Hinweise zum Erzeugen ähnlicher Animationssequenzen. (bl)

Editor 2x2 (Tastenfunktionen)

Taste	Aufgabe
<F1>	SET Pixelmodus an
<F3>	TOOGLE Pixelmodus an
<F5>	RESET Pixelmodus an
<F7>	COLOR MODUS an
<CLR/HOME>	Cursor in die linke obere Ecke
<SHIFT CLR/HOME>	Screen löschen und mit aktueller Farbe füllen
<DEL>	UNDO (macht letzte Aktion rückgängig)
<I>	bringt eine Liste aller Animationen im Directory
<L>	Animationssequenz laden. Die Tabelle klinkt sich ab der aktuellen Position ein. Die Pages oberhalb der gerade gültigen werden belegt.
<S>	Animation speichern
<CRSR links/rechts>	Bewegung in der Animationstabelle
<CRSR aufwärts/abw.>	in den Pages blättern
<CTRL CRSR aufwärts>	erhöht Verzögerung der aktuellen Page
<CTRL CRSR abwärts>	reduziert Verzögerung
<RUN/STOP>	Animationsende markieren = END
<Pfeil links>	Animation von Page 0 bis END abspielen
<SHIFT Pfeil links>	Animation ab aktueller Position bis END durchspielen
<>	Pages kopieren (Zielpage mit Cursor-Tasten einstellen, dann erneut <> drücken)
<1 bis 0>, <A bis F>	Farbe für COLOR-Modus setzen
<SHIFT 1 bis 0>, <SHIFT A bis F>	Hintergrundfarbe
<CBM 1 bis 0>, <CBM A bis F>	Menüfarben
<+/->	Farben für COLOR-MODUS erhöhen/reduzieren
<SHIFT +/->	... Hintergrundfarben
<CBM +/->	... Menüfarben

Kurzinfo: Editor 2x2/Player 2x2

Programmart: Grafik-Tool
Laden: LOAD "EDITOR 2X2",8
Starten: nach dem Laden RUN eingeben
Besonderheiten: benutzt die aus dem Blockgrafikzeichensatz bekannte Viertel-Byte-Grafik
Benötigte Blocks: 15
Programmautor: Andreas Schachtner

Char-Wandler V3

Der erste Eindruck ist der beste ...

Fetzige Vorspänne zu Spielen oder Anwendungsprogrammen – dazu braucht man unbedingt »Char-Wandler V3«, das ultimative Tool für jeden Demo-Programmierer!

Falls Sie sich schon gefragt haben, wie manche Freaks die unzähligen tollen Vorspänne oder Intros auf den Bildschirm zaubern: auch hier geht's nicht ohne vorbereitende Arbeit, um die notwendigen Grafikelemente zusammenzustellen. Wenn man aber auf derart komfortable Tools wie Char-Wandler V3 zurückgreift, läuft's wie geschmiert.

Laden Sie die Grafikanwendung mit:

```
LOAD "CHAR-WANDLER V3",8
```

Nach dem Start mit RUN kann man die Programmbeschreibung ausdrucken (Epson-kompatible, seriell angeschlossene Drucker) oder nur auf dem Bildschirm zu betrachten. <Z> zeigt die erste Seite der Anleitung, mit <+> und <-> blättert man weiter. Per <W> kommt man ins Intro. Erneuter Tastendruck aktiviert das Arbeitsmenü. Die Optionen lassen sich mit der entsprechenden Zifferntaste aufrufen:

<1> Char Turner

... verwandelt Hires- und Multicolorgrafiken in Zeichensätze. Folgende Bildformate akzeptiert das Programm: Amica Paint, Koala Painter, Paint Magic oder Hires (Hi-Eddi-Format). Zusätzlich stehen jede Menge Bearbeitungsfunktionen zur Verfügung (Grafik spiegeln, invertieren usw.). Per <X> geht's zurück ins Hauptmenü.

<2> Sprite Turner

... formt Sprite-Muster aus beliebigen Bereichen der Hires-Grafik (Abb. 2). In der linken oberen Bildschirmecke erscheint der Editor-Cursor, der sich per Joystick (Port 2) über den Bildschirm bewegen läßt. Das gewünschte Sprite-Muster übernimmt man per Feuerknopf oder <RETURN> ins RAM ab $4800 (18432). Per Plus- bzw. Minustaste blättert man im reservierten Sprite-Speicherbereich. Mit den Funktionstasten <F1> bis <F8> läßt sich das Sprite nachbearbeiten. Vor dem Speichern muß das zuletzt umgewandelte Sprite im Editorfenster (unterer Bildschirmrand) sichtbar sein! <F7> bringt Sie zurück ins Hauptmenü.

<3> Char to Hires

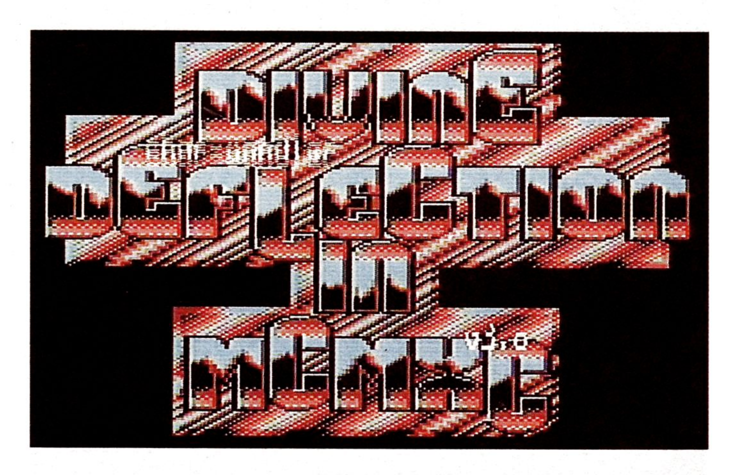
... macht aus beliebigen Zeichensatz-Grafiken Hires-Bilder. Zuerst lädt man den Font, dann den dazugehörenden Screen. Anschließend kann man die Hires-Grafik im Paint-Magic-, Koala-Painter- oder Hi-Eddi-Format speichern.

<4> Sprites to Hires

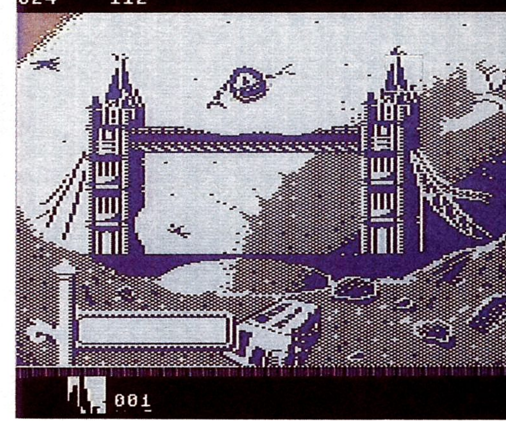
... wandelt eine Sammlung von Sprite-Mustern in Hires-Grafik um. Dazu muß man ebenfalls die entsprechende Sprite-Datei laden. Die Taste <C> leitet die Konvertierung ein, per <S> speichert man die Sprite-Muster als Hires-Grafik in den schon erwähnten Mal- und Zeichenprogrammformaten auf Disk.

<5> 1.1 Char Turner

... manipuliert den Original-Font des C 64 oder fremde Zeichensätze nach Belieben: vergrößern oder in x- und y-Richtung stauchen, drehen, spiegeln usw. Per <L> lädt man die gewünschten Zeichenmuster, <S> sichert sie auf Disk. Mit <X> geht's zurück ins Hauptmenü.



[1] Geänderte Zeichensätze und Multicolorgrafiken: Intro von Char-Wandler V3.



[2] Jeder Teilbereich einer Hires-Grafik läßt sich als Sprite-Muster definieren

<6> Hires to Scroll

Große, dreifarbige Zeichensätze (die man vorher per Malprogramm entworfen hat), lassen sich konvertieren, geordnet auf Disk speichern und in späteren Demos verwenden.

Die Zeichensatzgröße stellt man mit den Tasten <0> bis <F> ein (z.B. »3 x 2«, »4 x 4« etc.). Zusätzlich kann man die Mammutzeichen spiegeln, invertieren, drehen usw.

<7> Scroll to Hires

... dreht die Sache wieder um: die per Menüpunkt 6 transformierten, übergroßen Zeichensätze werden wieder in Hires-Grafik zurückverwandelt.

Geben Sie zunächst die Charset-Größe an (die bei Punkt 6 eingestellt wurde), dann die entsprechenden Dateinamen des Fonts und des Bildschirmgrafik-Files. Nach der Umwandlung läßt sich das Bild z.B. als Koala-Painter-Grafik sichern.

Programmroutinen für Demos und Intros, die vielfarbig über den Bildschirm flitzen (Abb. 1), müssen Sie allerdings selbst entwerfen – das kann Char-Wandler V3 nicht. Doch die dafür notwendigen Einzelgrafiken produziert man kaum noch komfortabler. (bl)

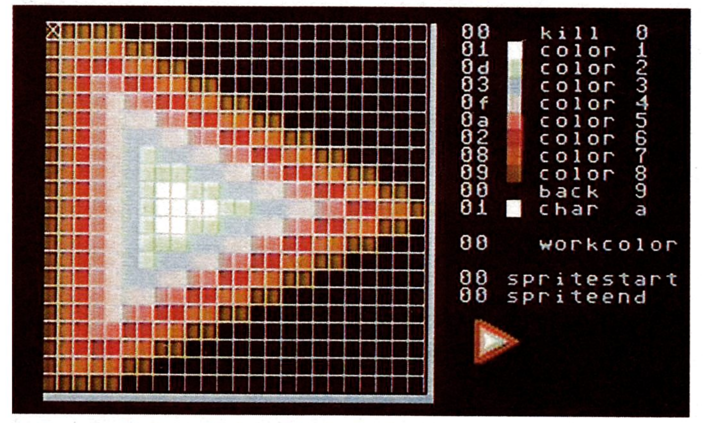
Kurzinfo: Char-Wandler V3

Programmart: Grafiktool
Laden: LOAD "CHAR-WANDLER V3",8
Starten: nach dem Laden RUN eingeben
Besonderheiten: berücksichtigt Grafiken der bekanntesten Mal- und Zeichenprogramme
Benötigte Blocks: 89
Programmautor: Sascha Lempke

Sprite Edit – 32 Sprites in acht Farben

Der Farben-Multi

Sprite-Editoren sind heute schon Massenware – aber keiner ist so farbträchtig wie unser Tool! »Sprite Edit« läßt bis zu acht verschiedene Farben pro Sprite zu. Sensationell:



Acht verschiedene Farben für ein Sprite-Muster

Mehr als vier Farben (inkl. Hintergrund) sind im Normalfall bei Multicolorsprites nicht einsetzbar. Möglich wird's erst durch die Programmiertricks unseres »Single-Sprite-Editors«.

Laden Sie das Programm mit:

```
LOAD "SPRITE EDIT (DD)",8
```

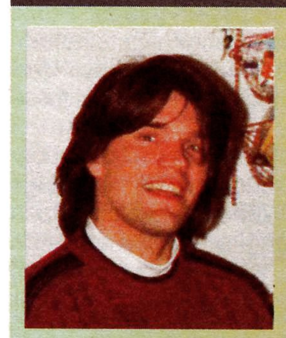
Nach dem Start mit RUN erscheint die Anleitung; per <SPACE> ruft man die nächste Bildschirmseite auf. In unserer Tabelle finden Sie eine Zusammenfassung aller Tastaturfunktionen des Sprite-Editors.

Wer das Programm sofort starten will, muß die Kombination <RUN/STOP RESTORE> drücken. Jetzt wählt man aus der Liste auf dem Screen, wie viele sich überlagernde Sprite-Muster fürs Editorfeld aktiviert werden sollen (1 bis 8). Unmittelbar darauf erscheint der Editorbildschirm (Abb.), rechts daneben das aktuell gültige Sprite-Muster in Originalgröße. Wem das ständige Flackern des Editorfeldes auf die Nerven geht, kann es mit <SHIFT C> abstellen.

Den Editor-Cursor bewegt man per Joystick Port 2 oder mit den Cursor-Tasten. Die gewünschten Sprite-Farben wählt man mit den Zifferntasten aus; Sprite-Pixel werden per Feuerknopf oder Leertaste gesetzt. Zum Löschen aktiviert man die Hintergrundfarbe (Taste <0>). Aktuell gültige Farben erkennt man in der Spalte »Workcolor«.

Mit den Tasten <+> und <-> blättert man vorwärts oder rückwärts im gesamten Sprite-Bereich (Offset $00 bis $1F, also maximal 32 Sprites). Achtung: nach dem Speichern belegt der gesamte Sprite-Speicher 65 Blocks auf Diskette! Betrachten Sie nach dem Programmstart doch die einzelnen Sprite-Muster von Sprite Edit (DD) – die zeigen den raffinierten Aufbau solcher Multicolor-Sprites am deutlichsten. (bl)

Sascha Lempke



... ist 23 Jahre alt und legte sich bereits 1981 einen der ersten C 64 zu.
Die Idee zu »Charwandler« beispielsweise entstand 1986, als er die Demo-Gruppe »CRYPT« gründete und begann, Intros und Demos mit dreifarbigen Logos und überdimensionalen Laufschriften zu programmieren. Da er sich darüber ärgerte, neue Charsets Zeichen für Zeichen entwerfen zu müssen, suchte er nach einem Weg, dies mit »Amica Paint« und »Hi-Eddi« zu realisieren. Schnell war das Programm »Charwandler 1.0« fertig, das Hires- und Mulitcolorgrafiken in Chars verwandelte.
Zusammen mit seiner Gruppe entwickelte er 1988 den »Charwandler 2.0«, der auch Scroll-Schrift berücksichtigte. 1989 hob man die verbesserte Version »Charwandler 3.0« aus der Taufe.
Inzwischen ist Sascha Lempke auf den Archimedes umgestiegen, und versucht, mit diesem RISC-Computer ebenfalls raffinierte Anwender- und Demoprogramme zu entwickeln.

Sprite Edit (DD) - Tastenfunktionen

Taste	Funktion
<F1>	speichert alle Sprite-Muster von »spritestart« bis »spriteend« auf Disk
<F2>	sichert Patterns (Sprite-Muster ohne Farbinformation) auf Disk
<F3>	lädt Sprites
<F4>	lädt Patterns
<F5>	Diskcommands (übliche DOS-Anweisungen ohne OPEN- und CLOSE-Befehle)
<F7>	Directory
<0 bis 8>	Einstellen der Zeichenfarbe. Nr. 0 ist stets die Löschfarbe.
<SHIFT 1 bis A>	färbt Pixelbereiche um
<CBM 1 bis 8>	Sprite als Pattern definieren
<CTRL 1 bis 8>	male mit Pattern 1 bis 8
<Q>	behält Sprite im Speicher
<SHIFT Q>	kopiert Sprite in anderen Musterbereich
<W>	Sprite um 90 Grad nach links drehen
<X>	horizontal spiegeln
<Y>	vertikal spiegeln
<SHIFT X>	Sprite in x-Richtung vergrößern
<SHIFT Y>	Sprite in y-Richtung vergrößern
<+/->	Sprite zur Bearbeitung wählen
<CLR HOME>	aktuelles Sprite-Muster löschen
<D>	Sprite nach unten bewegen
<SHIFT D>	bewegt Sprite ab Cursor-Position nach unten
<CBM D>	Pixelreihe ab Cursor-Position nach unten
<CTRL D>	Pixelreihe ab x/y-Position nach unten
<U>	Sprite nach oben,
<SHIFT U>	Sprite ab Cursor-Position nach oben
<CBM U>	Pixelreihe ab Cursor-Position aufwärts
<CTRL U>	Pixelreihe ab x/y-Position nach oben
<L>	Sprite nach links
<SHIFT L>	Sprite ab Cursor-Position nach links
<CBM L>	Pixelzeile nach links
<CTRL L>	Pixelzeile ab x/y-Position nach links
<R>	Sprite nach rechts
<SHIFT R>	Sprite ab Cursor-Position nach rechts
<CBM R>	Pixelzeile nach rechts,
<CTRL R>	Pixelzeile ab Cursor-Position nach rechts
<SHIFT C>	Rasterflash ein/aus
<SHIFT B>	Sprite-Raster ein/aus
<CRSR> oder Joystick Port 2	steuert Editier-Cursorblock
<SPACE> oder Feuerknopf	Punkt setzen

Kurzinfo: Sprite Edit (DD)

Programmart: Sprite-Editor-Tool
Laden: LOAD "SPRITE EDIT (DD)",8
Starten: nach dem Laden RUN eingeben
Besonderheiten: verteilt bis zu acht unterschiedliche Farben pro Sprite
Benötigte Blocks: 95
Programmautor: Sascha Lempke



[1] Im Meer versunken: London im Jahr 2697

Zukunftsvision?

Land unter!

Nichts ist unmöglich mit Malprogrammen wie »Amica Paint« oder »Koala Painter« – der Fantasie sind keine Grenzen gesetzt: man kann sogar London im Meer versinken lassen!

Weniger als düstere Zukunftsvision, sondern mehr als Gag-Grafik zu verstehen: Ende des 20ten Jahrhunderts setzte auf den Britischen Inseln ein Dauerregen ein, der Großbritannien langsam, aber sicher im Meer versinken ließ. Als sich im Jahr 2697 unweit westlich vor der dänischen Küste Tiefseetaucher tummelten, entdeckten sie auf dem Meeresgrund die fast unversehrt erhalten gebliebene Tower Bridge.

Abb. 1 zeigt die futuristische Szene. Die Grafik wurde mit Amica Paint entworfen und per Konverter-Utility ins Koala-Painter-Format umgewandelt. Es ist das letzte Bild der Koala-Show mit den Grafiken der Mondlandung. Wenn Sie das Bild mit dem Malprogramm »Koala-Painter« betrachten oder nachbearbeiten möchten, sollten Sie als Hintergrund- bzw. Rahmenfarbe stets schwarz wählen, damit sich die Wirkung des Bildes nicht verändert. Schwierig sind dunkle Farbübergänge zu realisieren, da der C 64 nur blau, dunkelgrau und braun als wirklich dunkle Farben anbietet. Übrigens: der rote Klotz neben dem Hinweisschild »Underground« ist ein umgestürzter Doppeldecker-Bus.

Um den Bezug zur Realität wiederherzustellen: London, wie wir's heute kennen, spiegelt sich in Abb. 2 in der Frontscheibe eines Taxis (Amica-Paint-Grafik). Abb. 1 wurde von Martin Lassahn entworfen, die zweite Grafik ist von Martin Kunz.

(bl)



[2] Typische Londoner Szene als Amica-Paint-Grafik

Koala-Painter-Diashow

Moonwalker

Beliebt wie eh und je bei den C-64-Grafik-Freaks: das Multicolor-Malprogramm »Koala-Painter«. Unser Leser Daniel Selinger hat damit eine Diashow zum Thema »Mondlandung« entworfen.



[1] Erinnerung an das Jahr 1969: Eroberung des Mondes



[2] Ramponierte Raumfähre billig abzugeben ...

[3] Die Erde geht auf ...

Das Hauptprogramm ist nicht nötig (obwohl die meisten Grafik-Fans Koala-Painter in der Diskettensammlung haben): unser Utility »Koala-Show 2.2« (wie jedes normale Basic-Programm zu laden und per RUN zu starten!) holt die Bilder von Diskette in den Speicher, blendet sie raffiniert ein und läßt sie auf dem Bildschirm stehen, bis man eine beliebige Taste drückt. Dann wird das nächste geladen.

Das sind die sechs Koala-Painter-Pictures auf Disk:
- Pic 1: Moon Walk (Abb. 1),
- Pic 2: For Sale (Abb. 2)
- Pic 3: Landing,
- Pic 4: Moon Base (Abb. 3),
- Pic 5: Space Disk,
- Pic 6: Crawler.

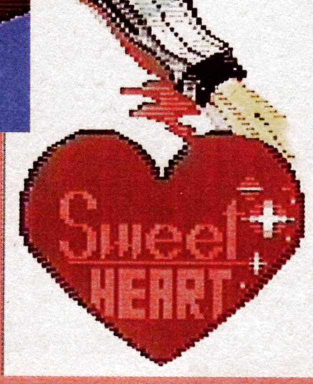
Viel Spaß beim Spaziergang auf dem Mond! (bl)

Fun Painter 2

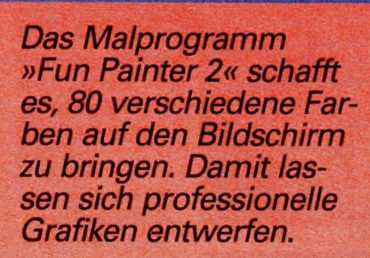
Neues vom Grafikmeister



[1] Toller Flitzer – der Ferrari 365 im Fun-Painter-Format



[2] Schreib' mal wieder (bevors Porto teurer wird!)



Das Malprogramm »Fun Painter 2« schafft es, 80 verschiedene Farben auf den Bildschirm zu bringen. Damit lassen sich professionelle Grafiken entwerfen.

Auf der Diskette zu diesem Sonderheft war kein Platz mehr, um das Malprogramm von Matthias Kranz nochmals zu veröffentlichen – wer's noch nicht hat, findet es auf der Programmservice-Diskette zum 64'er-Magazin 8/91.

Fun Painter 2 nutzt den Interlace-Modus: bei jedem Bildschirmaufbau wechseln die Farben – bedingt durch die Trägheit des menschlichen Auges meint man, Mischfarben zu sehen. Interlacing erzeugt Bildschirmflackern, das sich aber durch geschickte Farbwahl auf ein Minimum reduzieren läßt. Außerdem bedient sich das Programm der FLI-Technik (Flexible Line Interrupt): dieser spezielle Trick sorgt dafür, daß jedes Screen-Pixel eine eigene Farbe erhält (Ausnahme: die ersten drei Zeichen jeder Zeile).

Grafik-Ladeprogramm

Um Fun-Painter-Grafiken zu betrachten, braucht man nicht unbedingt das Hauptprogramm – dazu reicht ein Utility:

```
LOAD "FUNDISPLAY $1000",8
```

Nach dem Start mit RUN sind die Funktionen aktiv. Bilder lädt man jetzt mit einer SYS-Anweisung per Direkteingabe:

```
SYS 4096," Bildname"
```

Wichtig sind die beiden Hochpfeile zu Beginn des Grafiknamens – das Charakeristikum jeder Fun-Painter-Grafik.

Auf der Disk zu diesem Sonderheft finden Sie zwei Fun-Painter-Bilder: » ferrari 365« und » sweetheart". Peter Wodrig in Neubrandenburg hat sie kreiert. (bl)

SONDERHEFT 95 VORSCHAU

DISKETTE IM HEFT

C 128

Auch im Computerzeitalter der Multi-Performance-Chips »Pentium« oder «Alpha«, der Multitasking-Benutzeroberflächen wie »Windows 3.1« halten ihm unzählige Fans die Treue: der C 128 läßt sich nicht unterkriegen! Hier eine Auswahl der Highlights in unserem nächsten C-128-Sonderheft (Nr. 95):

- »Datengrafik«, das optimale Chart-Programm, verwandelt trockene Zahlen in übersichtliche Balken-, Torten- und Kurvendiagramme, die man selbstverständlich ausdrucken kann.
- »Mini-dBase«, die universelle Dateiverwaltung nach relationalem Muster, ist fast so komfortabel wie die großen Vorbilder der IBM-kompatiblen PCs/ATs!
- Sind Sie scharf auf die ultimative CP/M-Version 3.0 vom Juli 1987? Wir verraten Ihnen, wo's diese begehrte Software noch gibt und stellen neue CP/M-Programme vor!

Nr. 95 gibt's ab 28. 10. 93 bei Ihrem Zeitschriftenhändler

Aus aktuellen oder technischen Gründen können Themen ausgetauscht werden. Wir bitten dafür um Verständnis.

Jetzt will ich die Nr. 1

64'er

Das brandheiße Magazin für Computer-Fans gibt es jeden Monat superaktuell beim Händler oder am Kiosk.

Nix wie hin und

- die Tips&Tricks ins Haus geholt
- die neuesten Programme auf die Platte gebeamt
- ganze ausführliche Kurse durchgecheckt
- komplette Informationen zum C64er und C128er

Jeden Monat Gewinnchancen durch Wettbewerbe.

Jeden Monat das Neueste vom Spiele-Markt.



FOLGT SIE DEN BULLEN ODER LIEBER DOCH DEN BÄREN

Wie eng die Verflechtungen der internationalen Wirtschaftssysteme sind, hat die Weltöffentlichkeit auf spektakuläre Weise im Herbst 1987 erfahren.

Die Börsen von Tokio über Frankfurt bis nach New York standen über Wochen im Brennpunkt des Interesses, und täglich gab es neue, verwirrende Nachrichten aus den geheimnisvollen Nervenzentren der Weltwirtschaft.

Dabei ist die Grundregel für das gesamte Börsengeschehen doch so einfach: buy low, sell high.

Die Schwierigkeit liegt einzig darin, den richtigen Zeitpunkt zu treffen.

Wie man das macht, kann man im »Planspiel Börse« lernen, das die Sparkasse seit Jahren mit wachsender Beteiligung veranstaltet.

Mit fiktiven Einsätzen nehmen Jungbörsianer hier am realen Börsengeschehen teil, schöpfen Gewinne ab und kompensieren Verluste.

Die zunehmende Zahl begeisterter Teilnehmer hat auch diese Sparkassen-Initiative zum Erfolg geführt.

